LNAI 6119

Mohammed J. Zaki
Jeffrey Xu Yu
B. Ravindran
Vikram Pudi (Eds.)

Advances in Knowledge Discovery and Data Mining

14th Pacific-Asia Conference, PAKDD 2010
Hyderabad, India, June 2010
Proceedings, Part II

2 Part II

Springer

Mohammed J. Zaki   Jeffrey Xu Yu
B. Ravindran   Vikram Pudi (Eds.)

Advances in Knowledge Discovery and Data Mining

14th Pacific-Asia Conference, PAKDD 2010
Hyderabad, India, June 21-24, 2010
Proceedings
Part II

Springer

Volume Editors

Mohammed J. Zaki
Rensselaer Polytechnic Institute
Troy, NY, USA
E-mail: zaki@cs.rpi.edu

Jeffrey Xu Yu
The Chinese University of Hong Kong
Hong Kong, China
E-mail: yu@se.cuhk.edu.hk

B. Ravindran
IIT Madras, Chennai, India
E-mail: ravi@cse.iitm.ac.in

Vikram Pudi
IIIT, Hyderabad, India
E-mail: vikram@iiit.ac.in

Library of Congress Control Number: 2010928262

CR Subject Classification (1998): I.2, H.3, H.4, H.2.8, I.4, C.2

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN	0302-9743
ISBN-10	3-642-13671-0 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-13671-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

The 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining was held in Hyderabad, India during June 21–24, 2010; this was the first time the conference was held in India.

PAKDD is a major international conference in the areas of data mining (DM) and knowledge discovery in databases (KDD). It provides an international forum for researchers and industry practitioners to share their new ideas, original research results and practical development experiences from all KDD-related areas including data mining, data warehousing, machine learning, databases, statistics, knowledge acquisition and automatic scientific discovery, data visualization, causal induction and knowledge-based systems.

PAKDD-2010 received 412 research papers from over 34 countries including: Australia, Austria, Belgium, Canada, China, Cuba, Egypt, Finland, France, Germany, Greece, Hong Kong, India, Iran, Italy, Japan, S. Korea, Malaysia, Mexico, The Netherlands, New Caledonia, New Zealand, San Marino, Singapore, Slovenia, Spain, Switzerland, Taiwan, Thailand, Tunisia, Turkey, UK, USA, and Vietnam. This clearly reflects the truly international stature of the PAKDD conference.

After an initial screening of the papers by the Program Committee Chairs, for papers that did not conform to the submission guidelines or that were deemed not worthy of further reviews, 60 papers were rejected with a brief explanation for the decision. The remaining 352 papers were rigorously reviewed by at least three reviewers. The initial results were discussed among the reviewers and finally judged by the Program Committee Chairs. In some cases of conflict additional reviews were sought. As a result of the deliberation process, only 42 papers (10.2%) were accepted as long presentations (25 mins), and an additional 55 papers (13.3%) were accepted as short presentations (15 mins). The total acceptance rate was thus about 23.5% across both categories.

The PAKDD 2010 conference program also included seven workshops: Workshop on Data Mining for Healthcare Management (DMHM 2010), Pacific Asia Workshop on Intelligence and Security Informatics (PAISI 2010), Workshop on Feature Selection in Data Mining (FSDM 2010), Workshop on Emerging Research Trends in Vehicle Health Management (VHM 2010), Workshop on Behavior Informatics (BI 2010), Workshop on Data Mining and Knowledge Discovery for e-Governance (DMEG 2010), Workshop on Knowledge Discovery for Rural Systems (KDRS 2010).

The conference would not have been successful without the support of the Program Committee members (164), external reviewers (195), Conference Organizing Committee members, invited speakers, authors, tutorial presenters, workshop organizers, reviewers, authors and the conference attendees. We highly appreciate the conscientious reviews provided by the Program Committee members, and external reviewers. The Program Committee members were matched with the papers using the SubSift system (http://subsift.ilrt.bris.ac.uk/) for bid matching; we thank Simon Price and Peter Flach, of Bristol University, for developing this wonderful system. Thanks also to Andrei Voronkov for hosting the entire PAKDD reviewing process on the easychair.org site.

We are indebted to the members of the PAKDD Steering Committee for their invaluable suggestions and support throughout the organization process. We thank Vikram Pudi (Publication Chair), Pabitra Mitra (Workshops Chair), Kamal Karlapalem (Tutorials Chair), and Arnab Bhattacharya (Publicity Chair). Special thanks to the Local Arrangements Commitee and Chair R.K. Bagga, and the General Chairs: Jaideep Srivastava, Masaru Kitsuregawa, and P. Krishna Reddy. We would also like to thank all those who contributed to the success of PAKDD 2010 but whose names may not be listed.

We greatly appreciate the support from various institutions. The conference was organized by IIIT Hyderabad. It was sponsored by the Office of Naval Research Global (ONRG) and the Air Force Office of Scientific Research/Asian Office of Aerospace Research and Development (AFOSR/AOARD).

We hope you enjoy the proceedings of the PAKDD conference, which presents cutting edge research in data mining and knowledge discovery. We also hope all participants took this opportunity to share and exchange ideas with each other and enjoyed the cultural and social attractions of the wonderful city of Hyderabad!

June 2010

Mohammed J. Zaki
Jeffrey Xu Yu
B. Ravindran

PAKDD 2010 Conference Organization

Honorary Chair

Rajeev Sangal	IIIT Hyderabad, India

General Chairs

Jaideep Srivastava	University of Minnesota, USA
Masaru Kitsuregawa	University of Tokyo, Japan
P. Krishna Reddy	IIIT Hyderabad, India

Program Committee Chairs

Mohammed J. Zaki	Rensselaer Polytechnic Institute, USA
Jeffrey Xu Yu	The Chinese University of Hong Kong
B. Ravindran	IIT Madras, India

Workshop Chair

Pabitra Mitra	IIT Kharagpur, India

Tutorial Chairs

Kamal Karlapalem	IIIT Hyderabad, India

Publicity Chairs

Arnab Bhattacharya	IIT Kanpur, India

Publication Chair

Vikram Pudi	IIIT Hyderabad, India

Local Arrangements Committee

Raji Bagga (Chair)	IIIT Hyderabad, India
K.S. Vijaya Sekhar	IIIT Hyderabad, India
T. Ragunathan	IIIT Hyderabad, India
P. Radhakrishna	Infosys SET Labs, Hyderabad, India
A. Govardhan	JNTU, Hyderabad, India
R.B.V. Subramanyam	NIT, Warangal, India

Program Committee

External Reviewers

Chao Luo
Chen-Yi Lin
Chih jui Lin Wang
Chuancong Gao
Chun Kit Chui
Chun Wei Seah
Chun-Hao Chen
Chung An Yeh
Claudio Silvestri
Da Jun Li
David Uthus
De-Chuan Zhan
Delia Rusu
Dhruv Mahajan
Di Wang
Dinesh Garg
Elena Ikonomovska
En Tzu Wang
Eugenio Cesario
Feilong Chen
Feng Chen
Ferhat Ay
Gokhan Yavas
Gongqing Wu
Hea-Suk Kim
Hideyuki Kawashima
Hui Zhu Su
Ilija Subasic
Indranil Palit
Jan Rupnik
Janez Brank
Jeyashanker Ramamirtham
Jitendra Ajmera
Junjie Wu
Kathy Macropol
Khalil Al-Hussaeni
Kong Wah Wan
Krishna Prasad Chitrapura
Kunpeng Zhang
Kyle Chipman
L. Venkata Subramaniam
Lang Huo
Lei Liu
Lei Shi
Lei Yang
Leting Wu
Li Zheng
Li An
Li Da Jun
Lili Jiang
Ling Guo
Linhong Zhu
Lixin Duan
Lorand Dali
Luka Bradesko
Luming Zhang
Manas Somaiya
Mark Beltramo
Markus Ojala
Masayuki Okabe
Miao Qiao
Michael Barnathan
Min-Ling Zhang
Mingkui Tan
Mitja Trampus
Mohammed Aziz
Mohammed Imran
Muad Abu-Ata
Nagaraj Kota
Nagarajan Krishnamurthy
Narasimha Murty Musti
Narayan Bhamidipati
Ngoc Khanh Pham
Ngoc Tu Le
Nicholas Larusso
Nidhi Raj
Nikolaj Tatti
Ning Ruan
Nirmalya Bandyopadhyay
Nithi Gupta
Noman Mohammed
Palvali Teja
Pannagadatta Shivaswamy
Paolo Trunfio
Parthasarathy Krishnaswamy
Pedro P. Rodrigues
Peipei Li
Prahladavaradan Sampath
Prakash Mandayam Comare
Prasad Deshpande

Prithviraj Sen
Pruet Boonma
Qi Mao
Qiang Wang
Qingyan Yang
Quan Yuan
Quang Khoat Than
Rahul Chougule
Ramanathan Narayanan
Raquel Sebastiao
Rashid Ali
Rui Chen
S. Sathiya Keerthi
Shailesh Kumar
Sai Sundarakrishna
Saikat Dey
J. Saketha Nath
SakethaNath Jagarlapudi
Salim Akhter Chowdhury
Samah Fodeh
Sami Hanhijärvi
Satnam Singh
Sau Dan Lee
Sayan Ranu
Sergio Flesca
Shafkat Amin
Shailesh Kumar
Shalabh Bhatnagar
Shantanu Godbole
Sharanjit Kaur
Shazzad Hosain
Shenghua Gao
Shirish Shevade
Shirish Tatikonda
Shirong Li
Shumo Chu
Shuo Miao
Sinan Erten
Sk. Mirajul Haque
Soumen De
Sourangshu Bhattacharya
Sourav Dutta
Srinivasan Sengamedu
Srujana Merugu
Subhajit Sanyal
Sufyan Beg
Sugato Chakrabarty
Sundararajan Sellamanickam
Tadej Štajner
Takehiko Sakamoto
Thi Nhan Le
Tianrui Li
Timothy DeVries
Toshiyuki Amagasa
Venu Satuluri
Victor Lee
Vikram Pudi
Vishwakarma Singh
Viswanath G
Wang Wen-Chi
Wei Chu
Wei Jin
Wei Peng
Wei Su
Wendell Jordan-Brangman
Wenjun Zhou
Wenting Liu
Xiaofeng Zhu
Xiaogang Wu
Xin Liu
Xing Jiang
Xintian Yang
Xutong Liu
Yan Zhang
Yanchang Zhao
Yang Xiang
Yang Zhou
Yasufumi Takama
Yezhou Yang
Yilin Kang
Yin Zhang
Yong Ge
Yuan Liu
Yukai He
Yuko Itokawa
Zakia Sultana
Zubin Abraham

Organized by

IIIT Hyderabad, India

Sponsoring Institutions

AFOSR, USA

AOARD, Tokyo, Japan

ONRG, USA

Table of Contents – Part II

Table of Contents – Part I

Subclass-Oriented Dimension Reduction with Constraint Transformation and Manifold Regularization

Bin Tong and Einoshin Suzuki

Graduate School of Systems Life Sciences, Kyushu University, Japan
{bintong,suzuki}@i.kyushu-u.ac.jp

Abstract. We propose a new method, called Subclass-oriented Dimension Reduction with Pairwise Constraints (SODRPaC), for dimension reduction on high dimensional data. Current linear semi-supervised dimension reduction methods using pairwise constraints, e.g., must-link constraints and cannot-link constraints, can not handle appropriately the data of multiple subclasses where the points of a class are separately distributed in different groups. To illustrate this problem, we particularly classify the must-link constraint into two categories, which are the *inter-subclass must-link constraint* and the *intra-subclass must-link constraint*, respectively. We argue that handling the *inter-subclass must-link constraint* is challenging for current discriminant criteria. Inspired by the above observation and the cluster assumption that nearby points are possible in the same class, we carefully transform must-link constraints into cannot-link constraints, and then propose a new discriminant criterion by employing the cannot-link constraints and the compactness of shared nearest neighbors. For the reason that the local data structure is one of the most significant features for the data of multiple subclasses, manifold regularization is also incorporated in our dimension reduction framework. Extensive experiments on both synthetic and practical data sets illustrate the effectiveness of our method.

1 Introduction

In various applications, such as gene expression, image retrieval, etc., one is often confronted with high dimensional data [1]. Dimension reduction, which maps data points in a high-dimensional space into those in a low-dimensional space, is thus viewed as one of the most crucial preprocessing steps of data analysis. Dimension reduction methods can be divided into three categories, which are supervised ones [2], unsupervised ones[3], and semi-supervised ones[4]. The input data in these three categories are labeled data, unlabeled data, and both of them, respectively. In a typical real-world application, only a small number of labeled data points are available due to the high cost to obtain them [4]. Hence the semi-supervised dimension reduction may be considered to fit into the practical setting. Instead of labeled data points, some semi-supervised methods assume pairwise constraints, for it is easier for experts to specify them than to assign the class labels of data points. More specifically speaking, pairwise constraints consist of must-link constraints and cannot-link constraints. The pair of data points in a must-link constraint shares the same class label, while the pair of data points in a cannot-link constraint is given different class labels.

From another viewpoint, dimension reduction methods can be divided into nonlinear and linear ones. The former allows a nonlinear transformation in the mapping while the latter restricts itself to linear transformation. We consider a complex distribution of points that are distributed in multiple subclasses. In other words, the data points of one class form several separated clusters. A nonlinear method has a higher degree of freedom and hence can handle data with complex distribution effectively while a linear method tends to be incompetent in such a case.

In this paper, we restrict our attention to the linear semi-supervised dimension reduction for the data of multiple subclasses with pairwise constraints. Previously relevant methods [5] [6] [7] [8] implicitly assume that a class consists of a single cluster. If the points are of multiple subclasses, handling the pairwise constraints to project the points into multiple subclasses in the transformed space is challenging for linear dimension reduction. For a deep analysis, we particularly classify the must-link constraint into two categories. If two points in a must-link constraint reside in a single subclass, we define such a must-link constraint as an *intra-subclass must-link constraint*. On the contrary, if two points in a must-link constraint come from different subclasses, we define such kind of must-link constraint as an *inter-subclass must-link constraint*. We attribute the reason of the improper behavior of current linear methods to the fact that the *inter-subclass must-link constraint* most probably confuses the discriminant criteria of existing methods. The problem resulted from the *inter-subclass must-link constraint* is also encountered by the constraint transformation. For instance, a method in [9] transforms multiple must-link constraints, which are connected via points in two different classes, into a cannot-link constraint between the centroids of the points of two classes. This method fails to give a comprehensible meaning if the points belong to different subclasses because the centroids may fall into the region of another class.

To overcome above problems, we propose SODRPaC, which consists of two steps. In the first step, must-link constraints which satisfy several conditions are transformed into cannot-link constraints and the remaining must-link constraints are deleted. The idea behind this step is to reduce the harmfulness of the *inter-subclass must-link constraints* while exploiting the must-link constraint information as much as possible by respecting the cluster assumption [10]: nearby points on the same manifold structure in the original space are likely to belong to the same class. In the second step, we obtain a projection mapping by inventing a new discriminant criterion for dimension reduction, which is suitable for the data of multiple subclasses, and employing the manifold regularization [11], which is helpful for discovering the local structure of data.

2 Problem Setting and Motivation

The problem setting is defined as follows. We are given a set of N points $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, where $\mathbf{x}_i$ represents a point in a d-dimensional space, a set of must-link constraints $\mathbf{M} = \{m_1, m_2, \dots, m_{N_{\mathrm{ML}}}\}$, and a set of cannot-link constraints $\mathbf{C} = \{c_1, c_2, \dots, c_{N_{\mathrm{CL}}}\}$. Here m_i consists of a pair of points belonging to the same class while c_i consists of a pair of points belonging to different classes. The output is a

$d \times l$ transformation matrix W which consists of l projective vectors $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_l\}$ ($l \ll d$). W maps $\mathbf{x}_1$,$\mathbf{x}_2$,...,$\mathbf{x}_N$ to a set of lower dimensional points $Y = \{\mathbf{y}_1, ..., \mathbf{y}_N\}$. Hence $\mathbf{y}_i = W^T\mathbf{x}_i$ where $\mathbf{y}_i$ represents a point in a l-dimensional space. After making data projection, we only consider the classification task in the transformed space. For avoiding the bias caused by the choice of the classification method, the accuracy of nearest neighborhood (1-NN) classifier is considered as a measurement for the goodness of dimension reduction with the 20×5-fold cross validation.

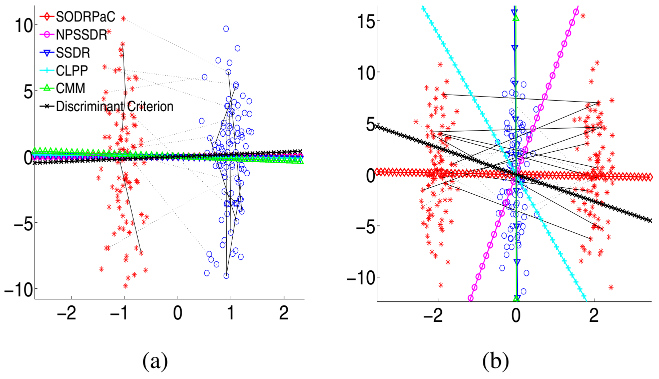


Fig. 1. Motivating examples. The data points are of Gaussian distribution. In (a), the blue and red points are distributed in different clusters. In (b), the red points reside in different subclasses. Must-link constraints and cannot-link constraints are denoted by black solid and dashed lines, respectively.

Fig. 1 presents the motivating examples, where $d = 2$ and $l = 1$. The task for dimension reduction here is thus to project the two dimensional data onto a line, where the points from different classes can be differentiated. A horizontal line is supposed to be the best projection while a vertical one is the worst projection. To better illustrate the motivation of our method, previously relevant methods are firstly retrospected. In the aspect of pairwise constraints, SSDR [5] and CMM [6] are to maximize the average distance between the points in cannot-link constraints, and to minimize the average distance between the points in must-link constraints simultaneously. We can see that minimizing the average distance between the points in must-link constraints is reasonable in the case shown in Fig. 1a, where all the must-link constraints are *intra-subclass must-link constraints*. However, it disturbs to maximize the average distance between the points in cannot-link constraints in the case shown in Fig. 1b, where all the must-link constraints are *inter-subclass must-link constraints*. CLPP [7] builds an affinity matrix, each entry of which indicates the similarity between two points. To utilize the constraint information, the affinity matrix is revised by setting the similarity degree between non-neighboring points involved in pairwise constraints. For example, given a must-link constraint, the similarity degree between two points is revised to be 1, indicating two points are close (similar) to each other, no matter the two points are distant (dissimilar) or not. Suppose that the must-link constraint is *inter-subclass must-link constraint*, it implies that the two points are not geometrically nearby each other. This arbitrary updating may damage the geometrical structure of data points. This problem is also confronted by NPSSDR [8]. The above analysis explains the reason why CMM, SSDR, CLPP and NPSSDR are capable of obtaining excellent performance as shown in Fig. 1a, while they fail to reach the same fine performance in the multiple subclass case shown in Fig. 1b.

In the light of observations, we argue that the *inter-subclass must-link constraint* is probably harmful for the discriminant criteria of existing methods. For this reason, we attempt to design a new discriminant criterion that is able to behave appropriately in the case of multiple subclasses. The new discrimination criterion marked as 'Discriminant Criterion' is able to obtain almost the same performance as others, as shown in Fig. 1a, and can even outperform previous methods, as shown in Fig. 1b. Moreover, the manifold regularization is helpful for discovering the local structure of data which is considered as one of the most principle characteristics of the data of multiple subclasses [12]. We therefore consider to make the new discriminant criterion and the manifold regularization work together in a collaborative way. Fig. 1b also demonstrates that our method SODRPaC, which is the combination of the new discrimination criterion and the manifold regularization, can obtain the best performance.

3 Subclass-Oriented Dimension Reduction with Pairwise Constraints

The overview of our SODRPaC involves two steps described as follows:

(1) ***Transformation***. This step transforms must-link constraints into cannot-link constraints under the cluster assumption.
(2) ***Dimension reduction***. This step includes two components. The first component is the new discriminant criterion suitable for the case of multiple subclasses. The other one is the manifold regularization, which helps discovering the local structure of data.

3.1 Transformation from Must-Link Constraints

Although a method that transforms must-link constraints into cannot-link constraints is provided in [9], we would point out that its purpose that the plentiful amount of constraints are reduced is substantially different from ours. Moreover, it becomes ineffective due to the *inter-subclass must-link constraint*. In a high dimensional space, the boundaries of subclasses and the number of subclasses within one class can not be explicitly detected by using the unlabeled data and link constraints. Thus, it is difficult to identify whether a must-link constraint is of *inter-subclass must-link constraint* or not. To reduce the harmfulness of *inter-subclass must-link constraints*, removing all the must-link constraints is, therefore, the most straightforward way. However, it can be regarded as a waste of must-link constraint information. Preserving the useful must-link constraints as much as possible in the form of cannot-link constraints is then the fundamental idea behind the transformation.

In our method, the transformation from must-link constraints into cannot-link constraints basically occurs when a must-link constraint and a cannot-link constraint are connected. Under the cluster assumption, it is natural to consider two nearby points as another form of must-link constraint, so that we have more opportunities to transform must-link constraints into cannot-link constraints. In this paper, we employ shared nearest neighbor (SNN) [13] to formulate the sense of 'nearby' points. A set of shared nearest neighbors is denoted by $\mathbf{N}_S = \{\mathbf{N}_S^{\mathbf{x}_1}, \mathbf{N}_S^{\mathbf{x}_2}, \ldots, \mathbf{N}_S^{\mathbf{x}_N}\}$ where $\mathbf{N}_S^{\mathbf{x}_i} = \{\{\mathbf{x}_i, \mathbf{x}_j\} | \mathbf{x}_i \in \mathbf{N}(\mathbf{x}_j), \mathbf{x}_j \in \mathbf{N}(\mathbf{x}_i)\}$. $\mathbf{N}(\mathbf{x}_i)$ denotes the k nearest neighbors set of $\mathbf{x}_i$. Let $|\mathbf{N}_S|$ be the number of the pairs of shared nearest neighbors, where $|\cdot|$ denotes the cardinality of a set. The value of SNN between $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as the number of points shared by their neighbors $SNN(i,j) = |\mathbf{N}(\mathbf{x}_i) \cap \mathbf{N}(\mathbf{x}_j)|$. The larger the value of SNN between two points is, the closer the two points are. It should be noted that we design a $N \times N$ matrix $\mathbf{L}$ to specify a kind of reliability for cannot-link constraints, which could be also deemed as the trustiness to them. Suppose that all the previously specified constraints are correct, for the previously given cannot-link constraints and the generated cannot-link constraints by using must-link constraints, their reliabilities are set to be 1. For the generated cannot-link constraints by using shared nearest neighbors, their reliabilities are equal to the similarities between the shared nearest neighbors. It is because that transformation by employing shared nearest neighbors are considered to be less trustful than that by using must-link constraints. We believe it is natural to take the similarity between the shared nearest neighbors as a measurement for the trustiness. For example, given a pair of shared nearest neighbors $\{\mathbf{x}_i, \mathbf{x}_j\}$, we represent the reliability of a generated cannot-link constraint by using it as a Gaussian kernel, which is a simple kernel and has been widely applied in research fields. The reliability is formulated as $\theta(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\gamma)$, where $\|\cdot\|$ denotes the Euclidian norm and γ is the kernel parameter. Note that, for the convenient access to the matrix $\mathbf{L}$, given a cannot-link constraint $c = \{\mathbf{x}_i, \mathbf{x}_j\}$, we use $\mathbf{L}(c)$ to denote the entries of $\mathbf{L}_{ij}$ and $\mathbf{L}_{ji}$, thus $\mathbf{L}$ is a symmetric matrix.

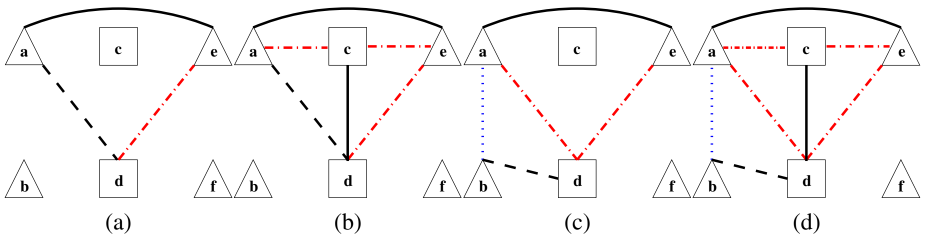


Fig. 2. Four simple cases for the transformation. The previously specified must-link constraints and cannot-link constraints are denoted by the black solid and dashed lines, respectively. The shared nearest neighbor is presented as the blue dotted line. The red dash-dotted line specifies the new cannot-link constraint.

Fig. 2 shows four fundamental scenarios of the transformation. The set $\{\mathrm{a}, \mathrm{b}, \mathrm{e}, \mathrm{f}\}$, and $\{\mathrm{c}, \mathrm{d}\}$ represent different classes of data points. We explain these four scenarios in a metaphorical way where the must-link constraint is taken as a friend relationship while the cannot-link constraint is considered as an enemy one. Standing from the viewpoint of point 'a', it is given a friend relationship, say $\{\mathrm{a}, \mathrm{e}\}$, as shown in Fig. 2a, which is called as a basic form. If ‘d’ is my enemy, instead of keeping my friend ‘e’, consider that ‘e’ is the enemy of my enemy ‘d’. Fig. 2b shows an extension of the basic form with an enemy’s friend rule. If my enemy ‘d’ has a friend ‘c’, ‘c’ is the enemy of my friend ‘e’ and me. In these two cases, the reliabilities for the new enemy relationships are set to be 1. Fig. 2c presents an extension of the basic form, which is called as a proximity form. If I have no enemy but my neighbor ‘b’ has an enemy ‘d’, ‘d’ is the enemy of my friend ‘e’ and me. Fig. 2d shows an extension of the proximity form with the enemy’s friend rule. Note that, in the latter two cases, the reliabilities for the new enemy relationships are set to be the similarity between my neighbor ‘b’ and me. The pseudo code for the summary of these four cases is illustrated in Algorithm 1.

Algorithm 1. Transformation from Must-link Constraints into Cannot-link Constraints

```
Input: $\mathbf{M}$, $\mathbf{C}$, $k$, $\gamma$.
Output: $\mathbf{C}$, $\mathbf{L}$.
 1: create a $N \times N$ zero matrix $\mathbf{L}$.
 2: for each $c \in \mathbf{C}$ do
 3:     $\mathbf{L}(c) = 1$.
 4: end for
 5: if $\exists\, c \in \mathbf{C}, m \in \mathbf{M}$ s.t. $m \cap c \neq \emptyset$ then
 6:     define $\mathrm{a} \in m \cap c$, $\mathrm{e} \in m - m \cap c$, $\mathrm{d} \in c - m \cap c$.
 7:     create a new cannot-link constraint $c' = \{\mathrm{d}, \mathrm{e}\}$; if $c' \notin \mathbf{C}$ then $\mathbf{C} \leftarrow \mathbf{C} \cup \{c'\}$, $\mathbf{L}(c') = 1$.
 8:     if $\exists\, m' \in \mathbf{M}$ s.t. $\mathrm{d} \in m'$, $m' \neq \{\mathrm{d}, \mathrm{e}\}$ then
 9:         define $\mathrm{c} \in m' - m' \cap c$.
10:         create two new cannot-link constraints $c'_1 = \{\mathrm{a}, \mathrm{c}\}$, $c'_2 = \{\mathrm{e}, \mathrm{c}\}$; for each $c'_i$, $i = 1, 2$,
            if $c'_i \notin \mathbf{C}$, then $\mathbf{C} \leftarrow \mathbf{C} \cup \{c'_i\}$, $\mathbf{L}(c'_i) = 1$.
11:     end if
12: end if
13: if $\exists\, m \in \mathbf{M}, c \in \mathbf{C}, \forall \mathrm{a} \in m, \forall \mathbf{n}_S^{\mathrm{a}} \in \mathbf{N}_S^{\mathrm{a}}$ s.t. $c \notin \mathbf{N}_S^{\mathrm{a}}$, $c \cap \mathbf{n}_S^{\mathrm{a}} \neq \emptyset$, $\mathrm{a} \notin c \cap \mathbf{n}_S^{\mathrm{a}}$ then
14:     define $\mathrm{d} \in c - c \cap \mathbf{n}_S^{\mathrm{a}}$, $\mathrm{e} \in m - m \cap c$.
15:     create two new cannot-link constraints $c'_1 = \{\mathrm{a}, \mathrm{d}\}$, $c'_2 = \{\mathrm{e}, \mathrm{d}\}$ and $r = \theta(\mathrm{a}, \mathrm{b})$; for
        each $c'_i$, $i = 1, 2$, if $c'_i \notin \mathbf{C}$, then $\mathbf{C} \leftarrow \mathbf{C} \cup \{c'_i\}$, $\mathbf{L}(c'_i) = r$.
16:     if $\exists\, m' \in \mathbf{M}$ s.t. $\mathrm{d} \in m'$ and $m' \neq \{\mathrm{d}, \mathrm{e}\}$ then
17:         define $\mathrm{c} \in m' - m' \cup c$.
18:         create two new cannot-link constraints $c'_1 = \{\mathrm{a}, \mathrm{c}\}$, $c'_2 = \{\mathrm{e}, \mathrm{c}\}$ and $r = \theta(\mathrm{a}, \mathrm{b})$; for
            each $c'_i$, $i = 1, 2$, if $c'_i \notin \mathbf{C}$, then $\mathbf{C} \leftarrow \mathbf{C} \cup \{c'_i\}$, $\mathbf{L}(c'_i) = r$.
19:     end if
20: end if
```

3.2 Dimension Reduction

In this section, we explain the dimension reduction which is based on a novel discriminant criterion and the manifold regularization. As mentioned in section 2, minimizing the average distance between the points in must-link constraints is inappropriate when the must-link constraints are *inter-subclass must-link constraints*. Under the cluster assumption, the shared nearest neighbors could be naturally deemed as another kind of *intra-subclass must-link constraints*. Thus, minimizing the average distance between the points in *intra-subclass must-link constraints* could be relaxed as making the shared nearest neighbors closer in the transformed space. Furthermore, the pair of points in the shared nearest neighbors probably resides in the same subclass, such that this relaxation would not suffer from the harmfulness of *inter-subclass must-link constraints*. Therefore, the discriminant criterion, which maximizes the average distance between the points in cannot-link constraints and minimizes the average distance between the shared nearest neighbors, is expected to be suitable for the data of multiple subclasses.

Suppose that $\mathbf{x}_i$ and $\mathbf{x}_j$ are projected to the image y_i^k and y_j^k along the direction $\mathbf{w}_k$, the new discriminant criterion is defined as follows:

$$\partial(\mathbf{w}_k) = \sum_{i,j:\{\mathbf{x}_i,\mathbf{x}_j\}\in\mathbf{C}} \mathbf{L}_{ij}\frac{\left\|y_i^k - y_j^k\right\|^2}{2|\mathbf{C}|} - \sum_{i,j:\{\mathbf{x}_i,\mathbf{x}_j\}\in\mathbf{N}_S} \mathbf{H}_{ij}\frac{\left\|y_i^k - y_j^k\right\|^2}{2|\mathbf{N}_S|} \tag{1}$$

where the elements of $\mathbf{H}$ are given below:

$$\mathbf{H}_{ij} = \begin{cases} SNN(i,j), & \{\mathbf{x}_i,\mathbf{x}_j\}\in\mathbf{N}_S^{\mathbf{x}_i} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Inspired by the local scatter [14], the intuition behind the latter part of the right side of Eq. 1 could be regarded as the compactness of shared nearest neighbors, since two points are more likely to be close if the value of SNN between them is large. The difference from the local scatter lies in the fact that a weighted matrix $\mathbf{H}$ which handles the similarity degree between shared nearest neighbors is employed. Since SNN provides a robust property that the side effect caused by the noisy points could be reduced to some degree, the compactness of shared nearest neighbors is more reliable than that of local scatter. The compactness of shared nearest neighbors could be also re-written as follows:

$$\begin{aligned}\sum_{i,j:\{\mathbf{x}_i,\mathbf{x}_j\}\in\mathbf{N}_S} \mathbf{H}_{ij}\frac{\left\|y_i^k - y_j^k\right\|^2}{2|\mathbf{N}_S|} &= \frac{1}{2|\mathbf{N}_S|}\sum_i\sum_j \mathbf{H}_{ij}(\mathbf{w}_k^T\mathbf{x}_i - \mathbf{w}_k^T\mathbf{x}_j)^2 \\ &= \mathbf{w}_k^T\left[\frac{1}{2|\mathbf{N}_S|}\sum_i\sum_j \mathbf{H}_{ij}(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^T\right]\mathbf{w}_k \\ &= \mathbf{w}_k^T S_1 \mathbf{w}_k \end{aligned} \tag{3}$$

where $S_1 = \frac{1}{2|\mathbf{N}_S|}\sum_i\sum_j \mathbf{H}_{ij}(\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^T$. S_1 then could be computed as follows:

$$\begin{aligned} S_1 &= \frac{1}{2|\mathbf{N}_S|}\left(\sum_i\sum_j \mathbf{H}_{ij}\mathbf{x}_i\mathbf{x}_i^T + \sum_i\sum_j \mathbf{H}_{ij}\mathbf{x}_j\mathbf{x}_j^T - 2\sum_i\sum_j \mathbf{H}_{ij}\mathbf{x}_i\mathbf{x}_j^T\right) \\ &= \frac{1}{|\mathbf{N}_S|}\left(\sum_i \mathbf{D}_{ii}\mathbf{x}_i\mathbf{x}_i^T - \sum_i\sum_j \mathbf{H}_{ij}\mathbf{x}_i\mathbf{x}_j^T\right) \\ &= \frac{1}{|\mathbf{N}_S|}\left(X\mathbf{D}X^T - X\mathbf{H}X^T\right) \end{aligned} \tag{4}$$

where $\mathbf{D}$ is a diagonal matrix whose entries are column sums of $\mathbf{H}$, $\mathbf{D}_{ii} = \sum_j \mathbf{H}_{ij}$. Similarly, the first part of right hand of Eq. 1 could be reformulated as:

$$\sum_{i,j:\{\mathbf{x}_i,\mathbf{x}_j\}\in\mathbf{C}} \mathbf{L}_{ij} \frac{\left\|y_i^k - y_j^k\right\|^2}{2|\mathbf{C}|} = \frac{1}{2|\mathbf{C}|} \sum_i \sum_j \mathbf{L}_{ij} (\mathbf{w}_k^T \mathbf{x}_i - \mathbf{w}_k^T \mathbf{x}_j)^2$$
$$= \mathbf{w}_k^T S_2 \mathbf{w}_k \tag{5}$$

where $S_2 = \frac{1}{|\mathbf{C}|}\left(X\mathbf{G}X^T - X\mathbf{L}X^T\right)$ where $\mathbf{G}$ is a diagonal matrix whose entries are column sums of $\mathbf{L}$, $\mathbf{G}_{ii} = \sum_j \mathbf{L}_{ij}$. Then, $\partial(\mathbf{w}_k)$ can be briefly written as:

$$\partial(\mathbf{w}_k) = \mathbf{w}_k^T X(\mathbf{P} - \mathbf{Q})X^T \mathbf{w}_k \tag{6}$$

where $\mathbf{P} = \mathbf{D} - \mathbf{H}$, and $\mathbf{Q} = \mathbf{G} - \mathbf{L}$. For all the $\mathbf{w}_k$, $k = 1, ..., l$, we can arrive at

$$\partial = tr\left[W^T X(\mathbf{P} - \mathbf{Q})X^T W\right] \tag{7}$$

where tr denotes the trace operator. As illustrated in Fig. 1b, the manifold regularization [11] is helpful for enhancing the performance obtained by the new discriminant criterion. We therefore incorporate it into our dimension reduction framework. The manifold regularization is defined as:

$$\xi = tr\left[W^T X\mathbf{M}X^T W\right] \tag{8}$$

where $\mathbf{M} = \mathbf{I} - \mathbf{K}^{-1/2}\mathbf{U}\mathbf{K}^{-1/2}$ is defined as a normalized graph Laplacian. $\mathbf{I}$ is a unit matrix, and $\mathbf{K}$ is a diagonal matrix whose entries are column sums of $\mathbf{U}$, $\mathbf{K}_{ii} = \sum_j \mathbf{U}_{ij}$, where $\mathbf{U}$ is defined as follows:

$$\mathbf{U}_{ij} = \begin{cases} \exp(\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2/\gamma), & \mathbf{x}_i \in \mathbf{N}(\mathbf{x}_j) \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

ξ is expected to be minimized in order to preserve the sub-manifold of data. At last, the final objective function that combines Eq. 7 and Eq. 8 together is expected to be maximized, and is derived as

$$\underset{\substack{W\in\mathbb{R}^{d\times l} \\ \text{s.t.} W^T W = \mathbf{I}}}{\arg\max} \; tr\left[W^T X(\mathbf{P} - \mathbf{Q} - \lambda\mathbf{M})X^T W\right] \tag{10}$$

where λ is a parameter to control the impact of manifold regularization. By introducing the Lagrangian, the objective function is given by the maximum eigenvalue solution to the following generalized eigenvector problem:

$$X(\mathbf{P} - \mathbf{Q} - \lambda\mathbf{M})X^T \mathbf{w} = \phi\mathbf{w} \tag{11}$$

where ϕ is the eigenvalue of $\mathbf{P} - \mathbf{Q} - \lambda\mathbf{M}$, and $\mathbf{w}$ is the corresponding eigenvector. One may argue that, when the graph of SNN is equal to the k-NN graph of the manifold regularization, $\mathbf{Q}$ is almost equivalent to $\mathbf{M}$ on preserving the local structure. As shown in [13], this situation would rarely happen since the two types of graph are dramatically different from each other in the general case. Moreover, to minimize the average distance between the shared nearest neighbors, which are considered as another form of must-link constraints, is conceptually distinct from preserving the local structure.

4 Evaluation by Experiments

4.1 Experiments Setup

We use public data sets to evaluate the performance of SODRPaC. Table 1 summarizes the characteristics of the data sets. All the data come from the UCI repository [15] except for GCM [16] that is of very high dimensionality. For the 'monks-1', 'monks-2', and 'monks-3' data, we combined the train and test sets into a whole one. For the 'letter' data, we chose 'A', 'B', 'C', and 'D' letters from the train and test sets respectively by randomly picking up 100 samples for each letter, and then assembled them into a whole set.

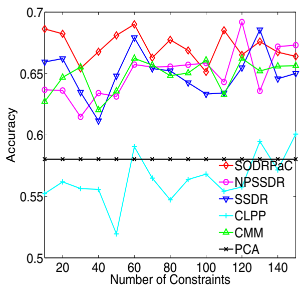
Table 1. Summary of the benchmark data sets

Data set	Dimension	Instance	Class	Data set	Dimension	Instance	Class
monks-1	6	556	2	monks-2	6	601	2
monks-3	6	554	2	letter(ABCD)	16	800	4
heart	13	270	4	GCM	16063	198	14

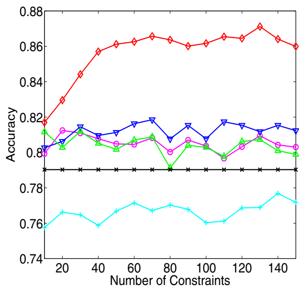
As shown in Eq. 10, λ is the parameter that controls the balance between $\mathbf{P}-\mathbf{Q}$ and $\mathbf{M}$. In this experiments setting, the parameter λ is searched from 2^{α}, where $\alpha \in \{\alpha| -5 \leq \alpha \leq 10, \alpha \in \mathbb{Z}\}$. A weighted 5-nearest-neighbor graph is employed to construct the manifold regularizer. In addition, the kernel parameter γ follows the suggestion in [17] that it is searched from the grid $\{\frac{\delta^2}{16}, \frac{\delta^2}{8}, \frac{\delta^2}{4}, \frac{\delta^2}{2}, \delta^2, 2\delta^2, 4\delta^2, 8\delta^2, 16\delta^2\}$, where δ is the mean norm of data. The parameter λ and the manifold regularizer are then optimized by means of the 5-fold cross-validation. As to the parameter settings of other competitive methods, we follow the parameters recommended by them, which are considered to be optimal. Without specific explanation, the number of must-link constraints is always set to be equal to that of cannot-link constraints, as the equal equilibrium between must-link constraints and cannot-link constraints is favorable for the existing methods. In addition, the value of k for searching shared nearest neighbors is set to be 3. The reason of this setting is to guarantee that the pairs of points in shared nearest neighbors reside in the same subclass, and to make the constraint transformation have more opportunities to be performed. In our experiments, must-link constraints and cannot-link constraints are selected according to the ground-truth of data labels.

4.2 Analysis of Experiments

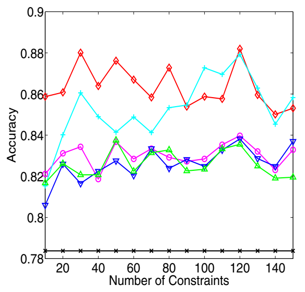
First, the effectiveness of SODRPaC is exhibited by changing the number of constraints. Apart from the semi-supervised dimension reduction methods, we also take PCA as the baseline. As illustrated in Fig. 3, SODRPaC always keeps the best performance when the number of available constraints increases from 10 to 150. As seen in Fig. 3a, Fig. 3b, Fig. 3d, and Fig.3f, CLPP is inferior to PCA even if the number of constraints is small. The side effect of *inter-subclass must-link constraints*, in this case, can be neglected.



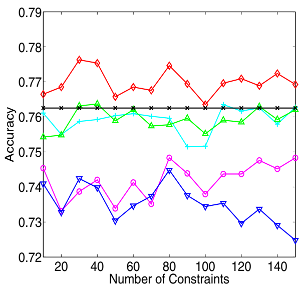
(a) monks-1 (d=2)



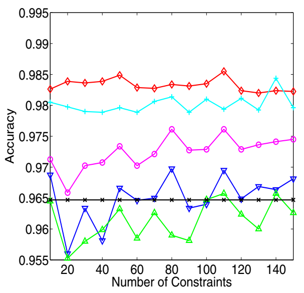
(b) monks-2 (d=4)



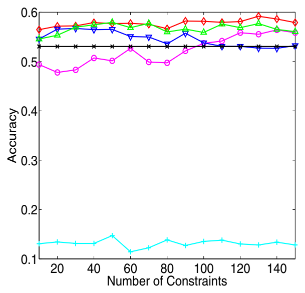
(c) monks-3 (d=5)



(d) heart (d=10)



(e) letter(abcd) (d=10)

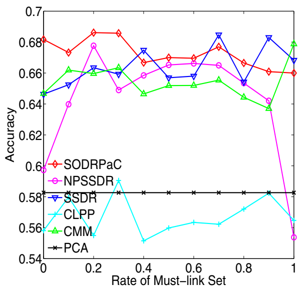


(f) GCM (d=150)

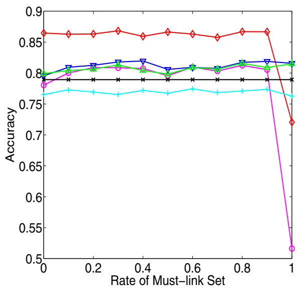
Fig. 3. The performance with different numbers of constraints (d: reduced dimensionality)

The reason is probably that the feature of discovering the local structure of data points could not help CLPP to outperform PCA. However, our SODRPaC, which also utilizes the manifold regularization due to its property of discovering the local structure, obtains the best performance. We can judge that the new discriminant criterion boosts the performance. It is also presented in Fig. 3d that the performance of SSDR decreases to some extent with the increase of the number of constraints. The possible reason is that increasing the number of available constraints makes the opportunity higher that *inter-subclass must-link constraints* exist, which deteriorates the optimization on the fine dimension reduction. It should be also pointed out that SODRPaC does not significantly outperform other methods. A possible reason is that the Euclidean distance, which is employed to formulate the similarity between points in the original space, is likely to be meaningless in the high dimensional space.

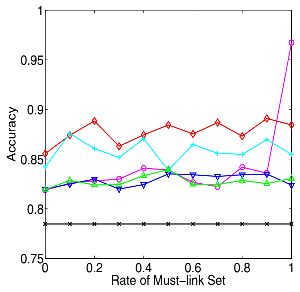
We then examine the relative impact between must-link constraints and cannot-link constraints on the performance of SODRPaC. In this experiment, given 150 available constraints, the ratio of must-link constraints to cannot-link constraints is varied from 0 to 1. Fig. 4 presents that SODRPaC has a much smoother behavior than others with the change of ratio. It indicates that SODRPaC is more robust than other semi-supervised methods in terms of the imbalance between must-link constraints and cannot-link constraints. As shown in Fig. 4b and Fig. 4f, SODRPaC presents an obvious degradation of



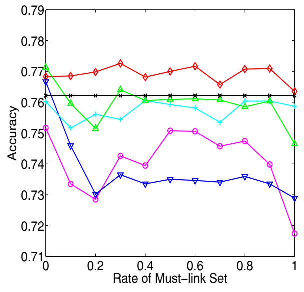
(a) monks-1 (d=2)



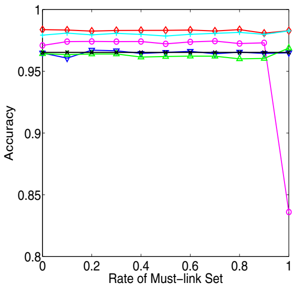
(b) monks-2 (d=4)



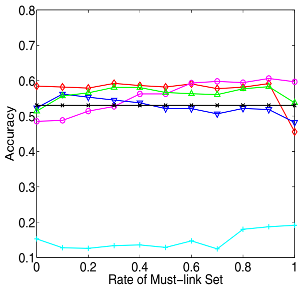
(c) monks-3 (d=5)



(d) heart (d=10)



(e) letter (abcd) (d=10)

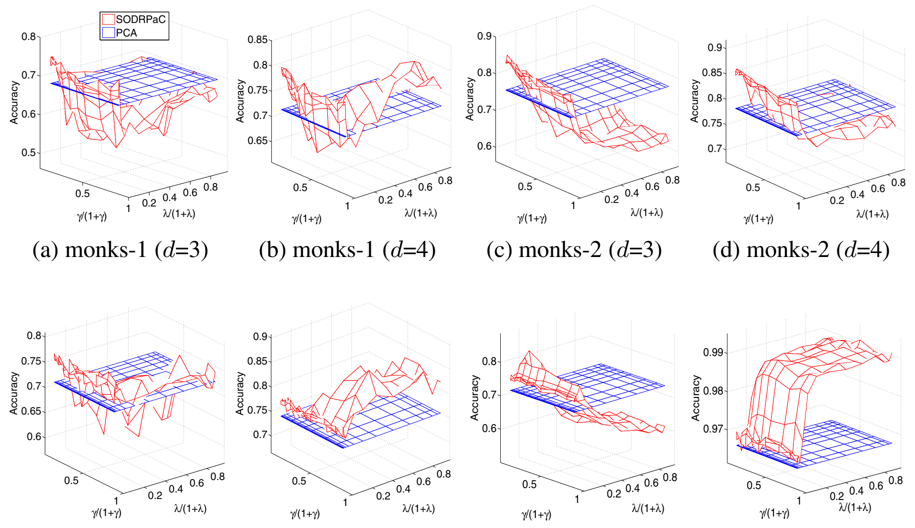


(f) GCM (d=150)

Fig. 4. The performance with the change of rate for must-link set (d: reduced dimensionality)

performance when all constraints are must-link ones. The most probable reason would be that the transformation from must-link constraints into cannot-link constraints can not be performed when the necessary cannot-link constraints lack. This behavior is consistent with the conclusion demonstrated in [9] that cannot-link constraints are more important than must-link constraints in guiding the dimension reduction.

As implicated in the previous sections, the parameter λ that controls the balance between $\mathbf{P} - \mathbf{Q}$ and $\mathbf{M}$, and the factor γ that is related to computing the similarity between two points would influence the performance of SODRPaC. An analysis on the two parameters is necessary to provide the guideline about how to choose their values. PCA is employed as the baseline because existing methods can not hold such two parameters simultaneously. Because of the different scale between λ and γ, λ-axis and γ-axis are thus plotted as $\lambda/(1+\lambda)$ and $\gamma/(1+\gamma)$, respectively. The axis values are then in the interval $(0, 1)$. We empirically uncover two interesting patterns for most of data sets and reduced dimensions as well. There are two regions where SODRPaC are more likely to obtain its best performance. The first region is where $\lambda/(1+\lambda)$ is small, as shown Fig. 5a, Fig. 5b, Fig. 5c, Fig. 5d, Fig. 5e and Fig. 5g. In this situation, the variation of $\gamma/(1+\gamma)$ would not cause the dramatic change for the performance of SODRPaC. The second region is where both $\lambda/(1+\lambda)$ and $\gamma/(1+\gamma)$ are large, as shown in Fig. 5b, Fig. 5e, Fig. 5f, and Fig. 5h.



(a) monks-1 (d=3) (b) monks-1 (d=4) (c) monks-2 (d=3) (d) monks-2 (d=4)

(e) monks-3 (d=3) (f) monks-3 (d=4) (g) letter(abcd) (d=3) (h) letter(abcd) (d=15)

Fig. 5. The analysis for λ and γ (d: reduced dimensionality)

5 Conclusions and Future Works

In this paper, we have proposed a new linear dimension reduction method with must-link constraints and cannot-link constraints, called SODRPaC, that can deal with the multiple subclasses data. Inspired by the observation that handling the *inter-subclass must-link constraint* is challenging for the existing methods, a new discriminant criterion is invented by primarily transforming must-link constraints into cannot-link constraints. We also combine the manifold regularization into our dimension reduction framework. The results of extensive experiments show the effectiveness of our method.

There are some other aspects of this work that merit further research. Although the empirical choice of λ and γ is suggested, we do not as yet have a good understanding of how to choose these two parameters which are also correlated with choice of the number of the reduced dimensionality. Therefore, we are interested in automatically identifying these three parameters and uncovering relationships among them. Another possible would be to integrate the semi-supervised dimension reduction and clustering in a joint framework with automatic subspace selection.

Acknowledgments. A part of this research is supported by the Strategic International Cooperative Program funded by Japan Science and Technology Agency (JST) and by the grant-in-aid for scientific research on fundamental research (B) 21300053 from the Japanese Ministry of Education, Culture, Sports, Science and Technology. Bin Tong is sponsored by the China Scholarship Council (CSC).

References

1. Parsons, L., Haque, E., Liu, H.: Subspace Clustering for High Dimensional Data: A Review. In: SIGKDD Explorations, pp. 90–105 (2004)
2. Fukunaga, K.: Introduction to Statistical Pattern Recognition. Academic Press, San Diego (1990)
3. Joliffe, I.: Principal Component Analysis. Springer, New York (1986)
4. Zhu, X.: Semi-supervised Learning Literature Survey. Technical Report Computer Sciences 1530, University of Wisconsin-Madison (2007)
5. Zhang, D., Zhou, Z.H., Chen, S.: Semi-supervised Dimensionality Reduction. In: Proceedings of the 7th SIAM International Conference on Data Mining (2007)
6. Wang, F., Chen, S., Li, T., Zhang, C.: Semi-supervised Metric Learning by Maximizing Constraint Margin. In: ACM 17th Conference on Information and Knowledge Management, pp. 1457–1458 (2008)
7. Cevikalp, H., Verbeek, J., Jurie, F., Klaser, A.: Semi-supervised Dimensionality Reduction Using Pairwise Equivalence Constraints. In: International Conference on Computer Vision Theory and Applications (VISAPP), pp. 489–496 (2008)
8. Wei, J., Peng, H.: Neighborhood Preserving Based Semi-supervised Dimensionality Reduction. Electronics Letters 44, 1190–1191 (2008)
9. Tang, W., Xiong, H., Zhong, S., Wu, J.: Enhancing Semi-supervised Clustering: A Feature Projection Perspective. In: Proc. of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 707–716 (2007)
10. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with Local and Global Consistency. In: Proc. Advances in Neural Information Processing Systems, pp. 321–328 (2004)
11. Belkin, M., Niyogi, P.: Manifold Regularization: A Geometric Framework for Learning From Labeled and Unlabeled Examples. Journal of Machine Learning Research 7, 2399–2434 (2006)
12. Sugiyama, M.: Dimensionality Reduction of Multimodal Labeled Data by Local Fisher Discriminant Analysis. Journal of Machine Learning Research 8, 1027–1061 (2007)
13. Ertöz, L., Steinbach, M., Kumar, V.: A New Shared Nearest Neighbor Clustering Algorithm and its Applications. In: Proc. of the Workshop on Clustering High Dimensional Data and its Applications, Second SIAM International Conference on Data Mining (2002)
14. Yang, J., Zhang, D., Yang, J.Y., Niu, B.: Globally Maximizing, Locally Minimizing: Unsupervised Discriminant Projection with Applications to Face and Palm Biometrics. IEEE Trans. Pattern Analysis and Machine Intelligence 29(4), 650–664 (2007)
15. Blake, C., Keogh, E., Merz, C.J.: UCI Repository of Machine Learning Databases, http://www.ics.uci.edu/~mlearn/MLRRepository.html
16. Ramaswamy, S., Tamayo, P., Rifkin, R., et al.: Multiclass Cancer Diagnosis Using Tumor Gene Expression Signatures. Proceedings of the National Academy of Sciences, 15149–15154 (1998)
17. He, X., Yan, S., Hu, Y., Niyogi, P.: Face Recognition Using Laplacianfaces. IEEE Transaction on Pattern Analysis and Machine Intelligence 27, 316–327 (2005)

Distributed Knowledge Discovery with Non Linear Dimensionality Reduction

Panagis Magdalinos, Michalis Vazirgiannis, and Dialecti Valsamou

Athens University of Economics and Business Athens, Greece
{pmagdal,mvazirg}@aueb.gr, dvalsamou@gmail.com

Abstract. Data mining tasks results are usually improved by reducing the dimensionality of data. This improvement however is achieved harder in the case that data lay on a non linear manifold and are distributed across network nodes. Although numerous algorithms for distributed dimensionality reduction have been proposed, all assume that data reside in a linear space. In order to address the non-linear case, we introduce D-Isomap, a novel distributed non linear dimensionality reduction algorithm, particularly applicable in large scale, structured peer-to-peer networks. Apart from unfolding a non linear manifold, our algorithm is capable of approximate reconstruction of the global dataset at peer level a very attractive feature for distributed data mining problems. We extensively evaluate its performance through experiments on both artificial and real world datasets. The obtained results show the suitability and viability of our approach for knowledge discovery in distributed environments.

Keywords: distributed non linear dimensionality reduction, distributed data mining.

1 Introduction

During the last decade, the evolution of the internet as well as the emergence of novel applications, such as peer-to-peer (P2P) systems, has led to an unprecedented information explosion. Information is distributed among network nodes, making the cost of centralizing and processing data prohibitive. Consequently, distributed data mining (DDM) has emerged as a highly challenging task.

Dimensionality reduction (DR) is an important step of data mining as high dimensional data lead to the degradation of query processing performance, a phenomenon known as the curse of dimensionality [8]. Thus typical tasks, such as clustering or classification, become ineffective. DR is then required in order to decrease the number of dimensions and reveal potentially interesting structures in data. With the advent of DDM, distributed dimensionality reduction (DDR) has emerged as a necessity in many applications.

A prominent such application is knowledge discovery from text collections distributed in a P2P network. Latest theoretical and experimental evidence point out that documents lay on a non linear high dimensional manifold ([5],[3]). Consequently, non linear dimensionality reduction (NLDR) is necessary in order to recover the low dimensional structure. Although numerous DDR algorithms have been proposed, all assume that data lay on a linear space. Thus there is the need for definition of distributed NLDR techniques.

To this end, we introduce *Distributed Isomap* (D-Isomap). D-Isomap corresponds to the decentralized version of the well known NLDR algorithm Isomap [18]. D-Isomap has been specifically designed and tuned in order to be applicable in large scale, structured P2P networks. We evaluate its performance and assess its viability and suitability for distributed environments through extensive experiments on artificial and real world datasets.

The contribution of this work is manifold. In section 2, we provide a review of the Isomap and DDR families of algorithms. In section 3 we introduce D-Isomap, a distributed NLDR algorithm which to the best of our knowledge is the first of its genre. Furthermore, we provide a cost model that assesses the computational and network resources required for the embedding of a dataset in a low dimensional space with D-Isomap. Finally, in section 4, we demonstrate the non linear nature of our approach through extensive experiments on well known non linear manifolds and justify its applicability in mining document collections distributed in P2P networks.

2 Related Work

DR algorithms are usually classified with respect to the way they manage data ([16]). *Linear algorithms* assume that high dimensional data lay on a linear or approximately linear manifold of significantly lower dimensionality. On the other hand, *non linear methods* assume that such linearity does not exist and operate on small fractions of the high dimensional manifold that can be perceived as locally linear. Due to space limitations, in the remaining of this section, we focus on the Isomap algorithm and its variations while in the end we provide a short overview of prominent DDR methods and motivate the need for a distributed NLDR approach.

Isomap [18] is a non linear technique that operates on points' pairwise geodesic distances. Isomap first constructs a nearest neighbor (NN) graph, where each point is represented as a node having edges to its k NN points. Edges are weighted according to the Euclidean distance of the points connecting. Global pairwise distances are calculated based on the shortest paths between all points (geodesic distances). The low dimensional mapping is derived by applying classic metric multidimensional scaling [19](MDS) on the geodesic distance matrix.

Isomap deficiencies to deal with curved manifolds or project large datasets gave rise to extensions such as C-Isomap and L-Isomap [16]. C-Isomap employs a different edge weighting scheme taking into account the mean distance of each point from its NNs. L-Isomap on the other hand attempts to address the excessive memory requirements of MDS by introducing Landmark MDS (LMDS). LMDS applies MDS on a set of sampled points and uses triangulation for the projection of the remaining dataset. Another problem of Isomap is the definition of non connected NN graphs. In such cases the algorithm operates on the largest connected component and discards the rest. A solution is provided by

Incremental Isomap [20] (I-Isomap) which guarantees the construction of a fully connected graph and is able to update the embedding when data is inserted or deleted.

DDR algorithms assume data distributed across a set of nodes and the existence of some kind of network organization scheme. The simplest case, where organization exists by construction, are structured P2P networks. In such networks, a protocol (usually based on distributed hast tables - DHT) ensures that any peer can efficiently route a search to a peer that has a specific file. Examples include Chord [17] and CAN [15]. In unstructured networks, the organization may be induced by means of physical topology (i.e. a router) or by means of self-organization [11]. In both cases however, a node undertakes all computations that have to be done centrally. The most prominent approaches in the area are adaptations of PCA ([9], [13], [14]). Two distributed alternatives of Fastmap [1] have also been proposed, but their application relies heavily on the synchronization of the network elements thus can only be applied in controllable laboratory environments. Recently, K-Landmarks [11] has appeared as a promising solution for DDR in unstructured P2P networks.

Unfortunately, all these methods are linear, in the sense that they assume that data lay on a linear or approximately linear low dimensional manifold. However, latest results point out that data usually lay on a non linear manifold ([5], [3]) thus linear methods fail to provide adequate results. Consequently, there is an apparent need for decentralized NLDR techniques. To the best of our knowledge, D-Isomap is the first attempt towards this direction.

3 Distributed Non Linear Dimensionality Reduction

D-Isomap capitalizes on the basic steps of Isomap and applies them in a network context, managing to successfully retrieve the underlying manifold while exhibiting tolerable network cost and computational complexity. In the rest of this section we present in details each step of the algorithm and review the cost induced by its application in a structured P2P network. Throughout the analysis we assume that N points, residing in R^d, are distributed in a P2P network of M peers. Each peer stores N_i points ($\sum_{i=1}^{M} N_i = N$). The objective is to recover the manifold residing in R^n using a graph defined by the k NNs of each point.

3.1 Data Indexing and Nearest Neighbours Retrieval

The first step of Isomap necessitates the definition of a kNN graph for each point. The latter, although applied in a distributed environment, should yield results of accuracy approximating that of a centralized approach. This, in conjunction with our initial goal for low network cost, highlights the need for a structured, DHT based, P2P network like Chord. Chord is a P2P lookup protocol where peer identifiers are arranged in a circle. Each node has a successor and a predecessor. The successor of a peer is the next node in the identifier circle when moving clockwise. On the other hand, the predecessor, is the next peer in the identifier circle when moving counter-clockwise. A message in Chord may require to traverse $O(logM)$ hops before reaching its destination.

In order to enable rapid lookup of points similar to each peer's local data we consider locality sensitive hashing [2] (LSH) that hashes similar points to the same bucket with high probability. LSH defines L hash tables, each related with a mapping function $g_i, i = 1...L$. Each g_i is defined by f hash functions $h_j(), j = 1...f$, randomly chosen from the same family of LSH functions $\mathcal{H}$. Every $h_{i,j}()$ maximizes the collision probability for data points that are close to each other in the original space. Since we measure similarity based on the euclidean distance, we use $h_{r,b}(x) = \lfloor \frac{rx+b}{w} \rfloor$, where x is the point under process, r is a random vector which coordinates are defined by the Gaussian distribution and w, b random numbers with the property $b \in [0, w)$.

The mapping of hash identifiers to peer identifiers is accomplished by employing a similarity preserving transformation that depicts a vector from R^f in R^1 [6]. For a given vector x, LSH produces an f-dimensional vector; the l_1 norm of this vector defines a similarity preserving mapping to R^1. Additionally, it can be proved that the obtained l_1 values are generated from the normal distribution $\mathcal{N}(\frac{f}{2}, \frac{f}{w}\mu_{l(x_i)})$, where $\mu_{l(x_i)}$ is the mean value of all points' Euclidean norm. Consequently, each hash value v is indexed by peer $p_i = (\frac{l_1(v)-\mu_{l_1}+2\sigma_{l_1}}{4*\sigma_{l_1}} * M)modM$.

The simplest way to retrieve the kNNs of a point p is to aggregate from all hash tables the points hashed in the same bucket as p. Afterwards, retrieve the actual points, calculate their distances from p and retain the kNNs. In order to reduce the required messages we halt the procedure as soon as ck points have been retrieved (in our experiments we set $c = 5$). Additionally, for each point, we define a range $bound_p$ that enables a queried peer to return only a subset of the points that indexes using Theorem 1. We use as bound the mean distance that a point exhibits from the points of its local dataset.

Theorem 1. *Given f hash functions $h_i = \lfloor \frac{r_i x^T + b_i}{w} \rfloor$ where r_i is an $1xn$ random vector, $w \in N$, $b_i \in [0, w), i = 1...f$, the difference δ of the l_1 norms of the projections x^f,y^f of two points x, $y \in R^n$ is upper bounded by $\frac{\| \sum_{i=1}^{f} |r_i| \| \|x-y\|}{w}$ where $\|x - y\|$ is the points' euclidean distace.*

Proof: Since $|a| - |b| \leq |a - b| \leq |a + b| \leq |a| + |b|$ we derive $l_1(x^f) \leq \frac{1}{w}\sum_{i=1}^{f}(|r_i x^T| + |b_i|) \leq \frac{1}{w}(\sum_{i=1}^{f}|r_i|)|x|^T + \frac{1}{w}\sum_{i=1}^{f}|b_i|$. We assume A=$(\sum_{i=1}^{f}|r_i|)$ and employ the inequality in order to derive $\delta = |l_1(x^f) - l_1(y^f)| \leq |\frac{1}{w}A(|x|^T - |y|^T)| \leq |\frac{1}{w}A||(|x|^T - |y|^T)| \leq |\frac{1}{w}A||(x^T - y^T)| = |\frac{1}{w}A||x - y|^T = \frac{1}{w}A|x - y|^T$ since w and $|r_i|$ are positive. In parallel, for any two vectors a, b we know that $\|ab^T\| \leq \|a\|\|b\|$. Consequently, $\delta \leq \frac{1}{w}\|A|x - y|^T\| \leq \frac{1}{w}\|A\|\|x - y\|$. Based on the latter we obtain $\delta \leq \frac{\|A\|\|x-y\|}{w}$ □

The first part of the procedure is presented in Algorithm 1. At first, Each peer, hashes its local points and transmits the derived l_1 values to the corresponding peers. This procedure yields a network cost of $O(N_i L)$ messages per peer or a total of $O(NL)$ messages. The process of recovering the kNNs of a point requires ck messages thus is upper bounded by $O(ckN)$. Time requirements on peer level

Algorithm 1. Data indexing and kNN retrieval

```
Input: Local dataset in $R^d(D)$, ♯ peers ($M$), ♯ hash tables $L$, hash functions $g$, ♯ NNs ($k$), peer identifier ($id$), parameter c ($c$)
Output: The local neighbourhood graph of peer $id$ ($X$)
for $i = 1$ to $N_{id}$, $j = 1$ to $L$ do
    $hash_j(p_i) = g_j(p_i)$ - where $p_i$ is the $i$-th point of $D$
    $peer_{ind} = (\frac{l_1(hash_j(p_i)) - \mu_{l_1} + 2\sigma_{l_1}}{4 * \sigma_{l_1}} * M) mod M$
    Send message $(l_1(hash_j(p_i)), id)$ to $peer_{ind}$ and store $(peer_{ind}, p_i, j)$
end for
if peer is creating its local NN graph then
    for $i = 1$ to $N_{id}$, $j = 1$ to $L$ do
        Send message $(id, hash_j(p_i), bound_{p_i})$ to $(peer_{ind}, p_i, j)$
        Wait for response message $(host, p_{ind}, l_1(p_{ind}))$
        If total number of received points is over $ck$, request points from host nodes, sort them according to their true distance from $p_i$ and retain the $k$ NNs of $p_i$
    end for
else
    Retrieve message $(id, hash_j(p_i), bound_{p_i})$ from $peer_{id}$
    Scan local index and retrieve relevant points according to Theorem 1
    Forward retrieved points' pointers to querying node
end if
```

are $O(N_i Lf + N_i klogk)$ induced by the hashing and ranking procedure. Finally memory requirements are $O(N_i k)$, due to retaining the NNs of each point.

3.2 Distributed Geodesic Distances Definition

Each point p has successfully recovered the location of its kNNs and created the corresponding graph G_p. Now, each peer should identify the shortest paths (SP) from its local points to all points in the dataset using only local information ($\cup_{j=1}^{N_i} G_j$). For this, we will use distributed SP computation techniques, extensively used for network routing purposes. A well known algorithm is the Distance Vector Routing (DVR) or Distributed Bellman-Ford (DBF) which is core part of many internet routing protocols such as RIP, BGP, ISO and IDRP [10].

For every point p, its host peer maintains a distance vector $DIST[p]$ that stores the distances from p to all points of the dataset. Initially, only the cells corresponding to p's NNs are populated while the rest are set to ∞. The procedure is initiated by sending the $DIST[p]$ to all peers maintaining p's NNs. Each receiver evaluates its current SPs to points appearing in $DIST[p]$ and if a new SP is identified updates distance vector $DIST[q]$ -where q is a point in p's set of NNs- and goes back to the sending step. The process is repeated until no update takes place, thus SPs have been computed.

The algorithm is asynchronous and does not have an explicit termination criterion. However it is self-terminating ([10]) since message transmissions will halt as soon as no updates take place. Consequently, each peer, after waiting time t to receive a message considers the process finalized. In order to guarantee the

Algorithm 2. Definition of geodesic distances

```
**Input:** peer id ($id$), local dataset ($D$), distances from NNs ($DIST$), time ($t$)
**Output:** SP distances of local points to the rest of the dataset ($DIST$)
**for** $i = 1$ to $N_i$ **do**
    Send $(DIST[i], i, id)$ to peers hosting NNs of $p_i$
**end for**
**while** Time to receive a message $< t$ **do**
    Receive message $(DIST[n], p, peer_j)$ - distances of $p_p$'s NN $n$ residing in $peer_j$
    **if** $d(p,j) > d(p,n) + d(n,j)$ for any $j \in DIST[n]$ **then**
        $DIST[p][j] = d(p,n) + d(n,j)$
    **end if**
    **if** update took place **then**
        Send $(DIST[p], p, id)$ to peers hosting NNs of $p_p$
    **end if**
**end while**
Substitute $\infty$ with $5 * max(DIST)$
```

creation of a connected SP graph we substitute in the end all remaining ∞ values with five times the largest local geodesic distance. Based on the description, the algorithm is presented in Algorithm 2.

DBF resembles a progressive range search where each point p learns in loop i distance information from points that are i edges away in the graph. Therefore, DBF execution requires $O(kDN^2)$ messages, where D is the diameter (the longest path) of the network. In our case, D depicts the number of peers that maintain parts of a single shortest path (from any point p to any point q), thus $D = M$ and the network cost is upper bounded by $O(kMN^2)$. Although the latter is relatively large, efficient implementation of the procedure can significantly reduce the total overhead. This can be accomplished by transmitting only updated SP information in the form $(p_{source}, p_{destination}, dist)$. Memory requirements are low, $O(N_iN)$ due to retaining the local points' distances in main memory throughout computations. Finally, time requirements are $O(M)$.

3.3 Approximating the Multidimensional Scaling

At this point, each peer has retrieved the SP distances of its own points to the rest of the dataset. The final step is to apply eigendecomposition on the global distance matrix, which is essentially the MDS step of Isomap. Although several methods for parallel computation of this procedure exist (i.e. [7]), they exhibit excessive network requirements, making their application infeasible. An approach that yields zero messages yet rather satisfactory results is the approximation of the global dataset at peer level with landmark based DR techniques. Instead of trying to map all data simultaneously to the new space, landmark-based DR algorithms use a small fraction of points and project them in the new space. Based on the assumption that these points remain fixed (landmarks in the new space), the rest of the dataset is projected using distance preservation techniques.

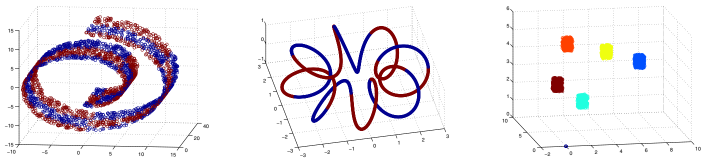
Algorithm 3. Distributed Isomap

```
Input: Local dataset in $R^d(D)$, ♯ peers ($M$), ♯ hash tables ($L$), hash functions ($g$), ♯ NNs ($k$), peer identifier ($p$), lower dimensionality ($n$), parameter c ($c$), aggregator peer ($p_a$), ♯ landmarks ($a$),time ($t$)
Output: The embedding of the global dataset in $R^n$ ($GL$)
Set $X$ = Define local neighbourhoods($D, M, L, g, k, p, c$) - Algorithm 1
Set $Y$ = Distributed Bellman-Ford($p, D, X, t$) - Algorithm 2
$LAN$ = local points (set of landmark points)
if $N_i < a$ then
  if $p <> p_a$ then
    Randomly select a subset of local points and transmit them to $p_a$
    Retrieve global landmarks $LAN$ from $p_a$
  else
    Receive $LAN_i$ from peer $i$
    Define $LAN$ by selecting $a$ landmarks and transmit it to all peers
  end if
end if
$GL$ = LMDS($Y$,$LAN$,$n$) or FEDRA ($Y$,$LAN$,$n$)
```

Two approaches directly applicable in our case are LMDS [16] and FEDRA [12]. LMDS operates by selecting a set of a landmark points, with the constraint $a > n$ and projects them in the new space with the use of MDS. Afterwards, a distance-based triangulation procedure, which uses as input distances to already embedded landmark points, determines the projection of the remaining points. FEDRA behaves similarly to LMDS however selects exactly n landmarks. LMDS requires $O(naN_i + a^3)$ time and $O(aN_i)$ space while FEDRA requires $O(nN_i)$ and $O(n^2)$ respectively. The salient characteristic of this step is that by using as landmarks the local points of a peer we manage to kill two birds with one stone. On one hand we embed the local dataset in R^n while simultaneously each peer derives an approximation of the global dataset. Consequently, each node is able to access global knowledge locally.

A potential failure may appear if the landmarks in a peer are not sufficient for the embedding to take place (i.e. for LMDS $a < n$). In such case, a network wide landmark selection process can be applied. The simplest way, is to assign a peer with the role of aggregator and then all peers transmit at least $\lceil \frac{n}{M} \rceil$ local points. Landmarks are randomly selected from the accumulated points and transmitted back to all nodes thus inducing $O(nNM)$ network load. Based on the previous analysis we derive D-Isomap and present it in Algorithm 3. The application of D-Isomap requires $O(N_iLf + N_iklogk)$ time and $O(n^2 + N_i(N + k))$ space per peer and a total of $O(NL + kMN^2)$ messages from all peers.

A final issue is related to the addition or deletion of data. Upon the arrival of a point, we apply Algorithm 1 and derive its kNNs. Afterwards, the SPs can be easily obtained using the fact that given a set of nodes in a graph, i.e $(s, n_1, n_2, ..., n_k, e)$, the distances from s to each n_i and from each n_i to e, the shortest path from s to e is the one minimizing the overall distance. Therefore, we relay on the retrieved k-NNs and calculate the SPs of the new point from all local



(a) Swiss Roll (b) Helix (c) 3D Clusters

Fig. 1. The Swiss Roll, Helix and 3D Clusters datasets

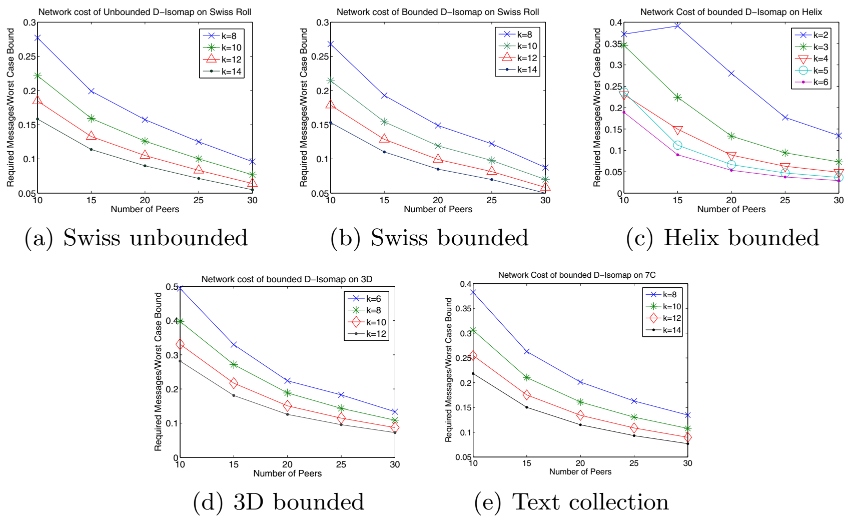
landmarks. Finally, we obtain its embedding through LMDS or FEDRA. The procedure requires $O(ck)$ messages. The case of deletion is much simpler, since the host node will only transmit message $(point_{id}, del)$ and force the deletion of the point from the bucket of the indexing peer. On the other hand, the arrival or departure of peers is handled by the Chord protocol itself.

4 Experiments

In this section we present the experimental evaluation of D-Isomap, which indeed verifies the expected performance and promotes it as an attractive solution for hard DDR and DDM problems. We carried out two types of experiments. First, we compared the manifold unfolding capability of D-Isomap in a distributed context against Isomap and L-Isomap in various network settings. In the second set of experiments, we evaluated D-Isomap's performance against Isomap, L-Isomap and LSI [4] in numerous supervised and usupervised DDM experiments using a medium sized text collection. The obtained results prove the suitability and viability of our algorithm for DDM problems, where each node holds a subset of the available information.

In the first set of experiments we employed three $3D$ non linear manifolds, namely the Swiss Roll, Helix and 3D Clusters each consisting of 3000 points (Figures 1(a), 1(b), 1(c)). In the case of the Swiss Roll an NLDR algorithm should unfold the roll into a parallelogram while in the case of Helix it should extract a circle. Concerning the 3D Clusters, we target in retaining the cluster structure in the new space. Each dataset was randomly distributed in a network of M peers ($M = 10, 15, 20, 25$ and 30). Depending on the dataset, we varied the value of k; for the Swiss Roll we set $k = 8$ and progressively augmented it by 2 until 14. For the 3D Clusters we started from $k = 6$ and reached 12 using the same step. For Helix we ranged k from 2 to 6 with a step of 1. In all experiments we set $c = 5$, $L = 10$, $f = 10$ and $w = 16$.

We assess the quality of D-Isomap by comparing the produced manifold (on peer level) against those produced centrally by Isomap and L-Isomap. For L-Isomap we set $a = 300$ in all experiments. In the subsequent graphs, D_F-Isomap or D_F indentifies D-Isomap configured with FEDRA and D_L-Isomap or D_L,



(a) Swiss unbounded (b) Swiss bounded (c) Helix bounded

(d) 3D bounded (e) Text collection

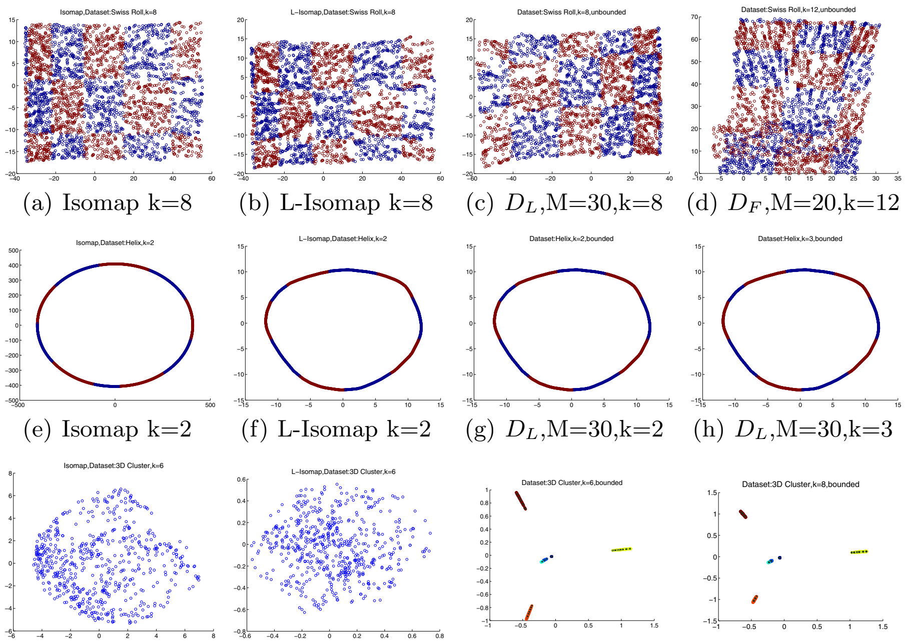
Fig. 2. Network cost reported as a fraction of the worst case bound $\frac{RequiredMessages}{WorstCaseBound}$

D-Isomap deployed with LMDS. We used MATLAB R2008a for the implementation of the algorithms and E2LSH [2] for LSH. Due to space limitations, we report only a subset of the experiments[1].

In Figure 2 we present the required number of messages for the projection of each dataset with D-Isomap as a fraction of the worst case network cost $(\frac{RequiredMessages}{WorstCaseBound})$ as derived by Section 3.3. First we validated the bound of Theorem 1 with the Swiss Roll. The results (Figures 2(a), 2(b)) indicate a reduction in the number of messages; consequently we employed the bounded version of the algorithm for all experiments. Figures 2(b), 2(c) and 2(d) provide the network load for the projection of each dataset with D-Isomap. The results highlight that D-Isomap behaves better in terms of network cost as the network size grows. The reason is simple; as the network grows, the buckets retained by each peer are smaller, therefore the messages are reduced. Moreover, messages are not affected seriously by changes in k so we observe a reduction in the percentage as k grows larger.

Figures 3(a), 3(b) depict the results obtained from Isomap and L-Isomap when applied on Swiss Roll for $k = 8$. Both algorithms have successfully revealed the underlying manifold. D_L-Isomap also recovered the correct 2D structure (Figure 3(c)) without being affected by the limited size of local data (only 3.3% of the global dataset). We report only one case of failure, for $M = 30$ and $k = 14$ where the embedding was skewed due to inappropriate selection of NNs. D_F-Isomap produces acceptable results however of lower quality compared to D_L-Isomap

[1] All experiments accompanied by the source code, the datasets and the graphs obtained from each peer are available at
`http://www.db-net.aueb.gr/panagis/PAKDD2010`



(a) Isomap k=8 (b) L-Isomap k=8 (c) D_L,M=30,k=8 (d) D_F,M=20,k=12

(e) Isomap k=2 (f) L-Isomap k=2 (g) D_L,M=30,k=2 (h) D_L,M=30,k=3

(i) Isomap k=6 (j) L-Isomap k=6 (k) D_L,M=30,k=6 (l) D_L,M=30,k=8

Fig. 3. Isomap, L-Isomap and D-Isomap on Swiss Roll (top), Helix (middle) and 3D Clusters (bottom)

(Figure 3(d)). This is due to the fact that FEDRA operates using only 2 points while LMDS employs the whole local dataset at each node.

Similar quality results were obtained from D_L-Isomap during the evaluation of Helix. Our algorithm managed to recover the circle structure of Helix (Figures 3(g), 3(h)) providing results comparable to L-Isomap (Figure 3(f)) and Isomap (Figure 3(e)). The inability of D_F-Isomap to work with a limited number of landmark points was more evident this time, producing an arc instead of a circle. The effectiveness of D-Isomap was proved when it was applied on the 3D Clusters dataset. Unlike Isomap and L-Isomap that failed to produce a connected graph (Figures 3(i), 3(j)), D_L-Isomap successfully managed to replicate the cluster structure in the new space (Figures 3(k)-3(l)) since by construction produces connected graphs. Again, D_F-Isomap failed to recover the whole cluster structure and preserved only three out of five clusters.

The results obtained from 3D Clusters inspired the application of D-Isomap on a DDM problem. As evaluation dataset, we used the titles of all papers published in ECDL, ECML/PKDD, FOCS, KDD, SIGMOD, SODA and VLDB conferences between 2006 and 2008 [2]. The dataset consists of 2167 papers, represented as 4726-dimensional vectors using a TF-IDF populated vector space model [4]. We randomly distributed the dataset among M peers ($M = 10$, 15,

[2] The authors would like to thank Dr. G. Tsatsaronis who kindly provided the dataset.

Table 1. Experiments on text. Is, LIs are used for Isomap,L-Isomap respectively

(a) Clustering experiments using k-Means

		Number of peers (M)												
		10		*15*		*20*		*25*		*30*				
		D_F	D_L	D_F	D_L	D_F	D_L	D_F	D_L	D_F	D_L	LSI	Is	LIs
	10	0.96	0.95	0.96	0.95	0.96	0.95	0.96	0.95	0.96	0.95	0.95	0.73	0.88
	15	0.96	0.95	0.96	0.95	0.96	0.95	0.96	0.95	0.96	0.95	0.98	0.70	0.85
Low Dim (n)	*20*	0.96	0.95	0.96	0.95	0.96	0.95	0.95	0.95	0.96	0.95	0.97	0.73	0.76
	25	0.95	0.95	0.95	0.95	0.96	0.95	0.95	0.95	0.95	0.95	0.90	0.71	0.81
	30	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.66	0.78

(b) Classification experiments using k-NN

		Number of peers (M)												
		10		*15*		*20*		*25*		*30*				
		D_F	D_L	D_F	D_L	D_F	D_L	D_F	D_L	D_F	D_L	LSI	Is	LIs
	10	1.11	1.04	1.10	1.07	1.11	1.08	1.10	1.08	1.10	1.08	1.14	0.84	0.76
	15	1.11	1.05	1.10	1.07	1.10	1.08	1.10	1.08	1.10	1.09	1.20	0.90	0.76
Low Dim (n)	*20*	1.10	1.05	1.10	1.07	1.10	1.08	1.10	1.09	1.10	1.09	1.24	0.90	0.80
	25	1.10	1.06	1.10	1.07	1.10	1.08	1.10	1.09	1.10	1.09	1.25	0.89	0.81
	30	1.10	1.06	1.10	1.07	1.10	1.08	1.10	1.09	1.10	1.09	1.24	0.90	0.79

20, 25 and 30) and embedded it in 10, 15, 20, 25 and 30 dimensions. We used the same values for L,f,a,c and w as before and ranged k from 8 to 14 with a step of 2. The embedded datasets from each peer were used as input for classification and clustering. We employed F-measure ([4]) in order to assess the quality of the results. In all experiments we report the relative quality amelioration $R = \frac{F_{m,new}}{F_{m,orig}}$. R represents the ratio of the F-measure ($F_{m,new}$) obtained in the low dimensional dataset over the F-measure ($F_{m,orig}$) obtained in the original case.

D_F-Isomap and D_L-Isomap were compared against Isomap, L-Isomap and LSI. For the central algorithms, reported values correspond to the mean of 10 executions. All results have been validated with a 10-fold cross validation. For D-Isomap we applied the same methodology on each peer level and report the mean value obtained across nodes. The statistical significance of D-Isomap's results has been verified by a t-test with confidence level 0.99. We employed k-Means and k-NN [4] for clustering and classification respectively; for k-NN we set $kNN = 8$ in all experiments. Although this may not be optimal, it does not affect our results, since we report the relative performance of the classifier.

Table 1(a) provides the clustering results obtained using $k = 8$ for the definition of the NNs for D-Isomap, Isomap and L-Isomap. The results highlight the applicability of D-Isomap in DDM problems as well as the non linear nature of text corpuses. Both flavours of our algorithm produce results marginally equal and sometimes superior to central LSI. The low performance of Isomap and

L-Isomap should be attributed to the definition of non-connected NN graphs. Table 1(b) provides the classification results obtained for the same value of k. D-Isomap is outperformed only by central LSI while in cases ameliorates the quality of k-NN. The latter comprises an experimental validation of the curse of dimensionality. The network load induced by D-Isomap in this experiment is provided in 2(e).

5 Conclusion

In this paper we have presented D-Isomap, a novel distributed NLDR algorithm which to the best of our knowledge is the first attempt towards this direction. We presented in details each step of the procedure and assessed the requirements posed to each network node by its application. Through extensive experiments we validated the capability of D-Isomap to recover linear manifolds from highly non linear structures. Additionally, we highlighted its applicability in DKD problems through experiments on a real world text dataset. The high quality results inspire us to pursue the extension D-Isomap towards P2P document retrieval and web searching.

References

1. Abu-Khzam, F.N., Samatova, N.F., Ostrouchov, G., Langston, M.A., Geist, A.: Distributed dimension reduction algorithms for widely dispersed data. In: IASTED PDCS, pp. 167–174 (2002)
2. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM 51(1) (2008)
3. Cai, D., He, X., Han, J.: Document clustering using locality preserving indexing. IEEE TKDE 17(12), 1624–1637 (2005)
4. Chakrabarti, S.: Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufmann, San Francisco (2002)
5. Gu, Q., Zhou, J.: Local relevance weighted maximum margin criterion for text classification. In: SIAM SDM, pp. 1135–1146 (2009)
6. Haghani, P., Michel, S., Aberer, K.: Distributed similarity search in high dimensions using locality sensitive hashing. In: ACM EDBT, pp. 744–755 (2009)
7. Henry, G., Geijn, R.: Parallelizing the qr algorithm for the unsymmetric algebraic eigenvalue problem. In: SIAM JSC, pp. 870–883 (1994)
8. Hinneburg, A., Aggarwal, C.C., Keim, D.A.: What is the nearest neighbor in high dimensional spaces? In: VLDB, pp. 506–515 (2000)
9. Kargupta, H., Huang, W., Sivakumar, K., Park, B.H., Wang, S.: Collective pca from distributed heterogeneous data. In: PKDD (2000)
10. Kurose, J.F., Ross, K.W.: Computer Networking: A Top-Down Approach Featuring the Internet. Addison-Wesley, Reading (2000)
11. Magdalinos, P., Doulkeridis, C., Vazirgiannis, M.: K-landmarks: Distributed dimensionality reduction for clustering quality maintenance. In: PKDD, pp. 322–334 (2006)
12. Magdalinos, P., Doulkeridis, C., Vazirgiannis, M.: Fedra: A fast and efficient dimensionality reduction algorithm. In: SIAM SDM, pp. 509–520 (2009)

13. Qi, H., Wang, T., Birdwell, D.: Global pca for dimensionality reduction in distributed data mining. In: SDMKD, ch. 19, pp. 327–342. CRC, Boca Raton (2004)
14. Qu, Y., Ostrouchov, G., Samatova, N., Geist, A.: Pca for dimensionality reduction in massive distributed data sets. In: 5th International Workshop on High Performance Data Mining (2002)
15. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: ACM SIGCOMM, pp. 161–172 (2001)
16. de Silva, V., Tenenbaum, J.B.: Global versus local methods in nonlinear dimensionality reduction. In: NIPS, pp. 705–712 (2002)
17. Stoica, I., Morris, R., Karger, D., Kaashoek, F.M., Hari: Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM (2001)
18. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290(5500), 2319–2323 (2000)
19. Togerson, W.S.: Theory and methods of scaling. Wiley, Chichester (1958)
20. Zhao, D., Yang, L.: Incremental isometric embedding of high-dimensional data using connected neighborhood graphs. IEEE TPAMI 31(1), 86–98 (2009)

DPSP: Distributed Progressive Sequential Pattern Mining on the Cloud

Jen-Wei Huang[1], Su-Chen Lin[2], and Ming-Syan Chen[2]

[1] Yuan Ze University, Taiwan
jwhuang@saturn.yzu.edu.tw
[2] National Taiwan University, Taiwan

Abstract. The progressive sequential pattern mining problem has been discussed in previous research works. With the increasing amount of data, single processors struggle to scale up. Traditional algorithms running on a single machine may have scalability troubles. Therefore, mining progressive sequential patterns intrinsically suffers from the scalability problem. In view of this, we design a distributed mining algorithm to address the scalability problem of mining progressive sequential patterns. The proposed algorithm DPSP, standing for Distributed Progressive Sequential Pattern mining algorithm, is implemented on top of Hadoop platform, which realizes the cloud computing environment. We propose Map/Reduce jobs in DPSP to delete obsolete itemsets, update current candidate sequential patterns and report up-to-date frequent sequential patterns within each POI. The experimental results show that DPSP possesses great scalability and consequently increases the performance and the practicability of mining algorithms.

1 Introduction

Based on the earlier work [7], the sequential pattern mining problem [1] can be categorized as three classes according to the management of corresponding databases. They are static sequential pattern mining, incremental sequential mining and progressive sequential pattern mining. It is noted that the progressive sequential pattern mining is known as a general model of the sequential pattern mining. The static and the incremental sequential pattern mining can be viewed as special cases of the progressive sequential pattern mining. The progressive sequential pattern mining problem can be described as "Given an interesting time period called period of interest (POI) and a minimum support threshold, find the complete set of frequent subsequences whose occurrence frequencies are greater than or equal to the minimum support times the number of sequences having elements in the current POI in a progressive sequence database." In fact, mining progressive sequential patterns intrinsically suffers from the scalability problem. In this work, we propose a distributed data mining algorithm to address the scalability problem of the progressive sequential pattern mining. The proposed algorithm DPSP, which stands for Distributed Progressive Sequential Pattern mining algorithm, is designed on top of Hadoop platform [6], which implements Google's Map/Reduce paradigm [5].

We design two Map/Reduce jobs in DPSP. At each timestamp, the candidate computing job computes candidate sequential patterns of all sequences and updates the summary of each sequence for the future computation. Then, using all candidate sequential patterns as the input data, the support assembling job accumulates the occurrence frequencies of candidate sequential patterns in the current POI and reports frequent sequential patterns to users. Finally, all up-to-date frequent sequential patterns in the current POI are reported. DPSP not only outputs frequent sequential patterns in the current POI but also stores summaries of candidate sequential patterns at the current timestamp. As time goes by, DPSP reads back summaries of all sequences and combine them with newly arriving itemsets to form new candidate sequential patterns at the new timestamp. Obsolete candidate sequential patterns are deleted at the same time. DPSP is thus able to delete obsolete itemsets, update summaries of all sequences and report up-to-date frequent sequential patterns. It is noted that DPSP does not need to scan the whole database many times to gather occurrence frequencies of candidate sequential patterns. DPSP, instead, reads newly arriving data and the summary of each sequence once. In addition, DPSP utilizes cloud computing techniques. It is easy to scale out using Hadoop platform to deal with huge amounts of data. The experimental results show that DPSP can find progressive sequential patterns efficiently and DPSP possesses great scalability. The distributed scheme not only improves the efficiency but also consequently increases the practicability.

The rest of this work is organized as follows. We will derive some preliminaries in Section 2. The proposed algorithm DPSP will be introduced in Section 3. Some experiments to evaluate the performance will be shown in Section 4. Finally, the conclusion is given in Section 5.

2 Related Works

After the first work addressing the sequential pattern mining problem in [1], many research works are proposed to solve the static sequential pattern mining problem [2], and the incremental sequential pattern mining problem [10]. As for the progressive sequential pattern mining problem, new data arrive at the database and obsolete data are deleted at the same time. In this model, users can focus on the up-to-date database and find frequent sequential patterns without being influenced by obsolete data. To deal with a progressive database efficiently, a progressive algorithm, Pisa, is proposed in [7]. However, traditional algorithms running on a single processor struggle to scale up with huge amount of data. In view of this, many researchers work on distributed and parallel data mining algorithms [4] [3] [12] [9] [11] [8]. In recent days, many researchers and corporations work on developing the cloud computing technology, which utilizes clusters of machines to cope with huge amount of data. The platform allows developers to focus on designing distributed algorithms whereas routine issues like data allocation, job scheduling, load balancing, and failure recovery can be inherently handled by the cloud computing framework. Hadoop [6] is an open

```
Algorithm DPSP
Input: itemsets of all sequences arriving at the current timestamp
1. while(there is new data at timestamp t){
2.    CandidateComputingJob;
3.    SupportAssemblingJob;
4.    t = t + 1;
5.    output frequent sequential patterns;
6. }end while
output: frequent sequential patterns at each POI with their supports
End
```

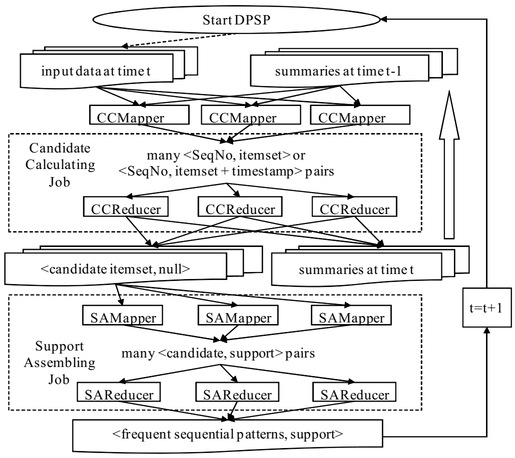


Fig. 1. Algorithm DPSP and system model

source project aiming at building a cloud infrastructure running on large clusters, which implements Google's Map/Reduce paradigm [5]. By means of the map function, the application can be divided into several fractions. Each fraction is assigned to a single node in large clusters and executed by the node. After the execution, the reduce function merges these partial results to form the final output. As such, developers need only to design a series of Map/Reduce jobs to split data and merge results.

3 Distributed Progressive Sequential Pattern Mining

We utilize Hadoop platform to design a distributed algorithm for the progressive sequential pattern mining. The proposed algorithm is named as Distributed Progressive Sequential Pattern mining algorithm, abbreviated as DPSP. In essence, DPSP consists of two Map/Reduce jobs, the candidate computing job and the support assembling job. As shown in the left of Figure 1, for each timestamp, the candidate computing job reads input data, which arrives at timestamp t, of all sequences. Itemsets from different sequences are distributed to different nodes in the cloud computing environment. Each node in the cloud computes candidate sequential patterns of each sequence within the current POI. Meanwhile, the candidate computing job also updates the summary for each sequence. Obsolete data are deleted in the candidate computing job and the up-to-date candidate sequential patterns are output. Then, support assembling job reads all candidate sequential patterns as input data. Different candidate sequential patterns are distributed to different nodes. Each node accumulates the occurrence frequencies of candidate sequential patterns and reports frequent sequential patterns whose supports are no less than the minimum support threshold in the current POI to users. When time goes to the next timestamp, DPSP keeps executing these Map/Reduce jobs. As such, DPSP is able to report the most up-to-date frequent sequential patterns in each POI.

The system model of DPSP is shown in right of Figure 1. The upper part is the candidate computing job while the support assembling job is at the lower part. In the candidate computing job, input data at timestamp t and the candidate set summaries at timestamp t-1 are split and transferred to several CCMappers. CCMapper generates many pairs of <sequence number, input itemset>. Then, pairs with the same sequence number are sent to the same CCReducer. CCReducer computes candidate sequential patterns of the given sequence and outputs pairs of <candidate sequential patterns, null>. In addition, CCReducer updates the summary of each sequence and deletes obsolete data at the same time. Candidate set summaries at the current timestamp are output for the computation at the next timestamp as well. Next, each SAMapper in the support assembling job reads input data and accumulates local occurrence frequencies for each candidate sequential patterns. SAMapper generates pairs of <candidate sequential pattern, local supports of the candidate> as outputs. Then, the pairs containing the same candidate sequential pattern are sent to the same SAReducer. SAReducer aggregates supports of the same candidate sequential pattern and outputs those frequent patterns in the current POI. After the computation at the timestamp, t, DPSP moves to the next timestamp, t+1.

3.1 Candidate Computing Job

The objective of the candidate computing job is to compute all candidate sequential patterns from all sequences within the current POI as shown in Figure 2. In CCMapper, itemsets of all sequences arriving at the current timestamp and the candidate set summaries at the previous timestamp are used as input data. As shown in lines 2 to 3 of CCMapper, if CCMapper reads the input from candidate set summaries, CCMapper generates <sequence number, candidate itemset with the corresponding timestamp> pairs. On the other hand, if CCMapper reads the input data from a sequence, CCMapper outputs <sequence number, arriving itemset> pairs as shown in lines 4 to 5. These output pairs are distributed to CCReducers as their inputs. Pairs with the same key are sent to the same CCReducer. By means of the summary at the previous timestamp and the arriving itemset at the current timestamp, each CCReducer is able to generate candidate sequential patterns of each sequence in the current POI. In line 2 of CCReducer, the multiple output variable is used to output candidate set summary at the current timestamp for the future computation. In lines 6 to 15, CCReducer enumerates each value in the receiving pairs. If the value is a candidate set summary at the previous timestamp, CCReducer puts the candidate into cand_set. In lines 9 to 10, if the timestamp is bigger than the start time of the current POI, which means this candidate will still be valid at the next timestamp, CCReducer outputs the candidate in the summary of the current timestamp for the computation at the next timestamp. In lines 11 to 12, if the candidate contains more than 1 item, the candidate is put in the result set as

```
CandidatesComputingJob:

CCMapper
input: itemsets of all sequences arriving at the current
timestamp OR candidate set summaries at the previous
timestamp
map:
1. var data = read input data;
2. if(data contains timestamp){ // it is a candidate
3.   output <data.sequenceNo, data.itemset+ data.time>;
4. else  // it is an arriving itemset
5.   output <data.sequenceNo, data.itemset>;

CCReducer
configure:
1. var start_time = current timestamp – POI;
2. var mo.output<sequenceNo, itemset + timestamp>;
   // multiple output to output summary
reduce(in_key, in_values):
3. var input; // used to store input data
4. var cand_set; // used to store candidate set summary at
the previous timestamp
5. var result<itemset>; // used to store distinct results
6. for(each value in in_values){
7.   if(value is a summary){
8.     cand_set.put(value.time, value.itemset);
9.     if(value.time > start_time)
10.      mo.output<in_key, candidate + candidate.time>;
11.    if(value.itemset.size > 1)
12.      result.put(value.itemset);
13.  }else // it is input data
14.    input = get the only one data at current timestamp;
15.}end for
16.for(each combination of items in input.itemset){
17.  for(each candidate in cand_set){
18.    var new_cand = append combination to candidate;
19.    result.put(new_cand);
20.    if(candidate.time > start_time)
21.      mo.output<in_key, new_cand + candidate.time>;
22.  }end for
23.  mo.output<in_key, combination + current time>;
24.}end for
25.for(each itemset in result)
26.  output<itemset, null>;
output: sequence number and candidate sequential itemsets
with timestamps

SupportsAssemblingJob:

SAMapper
input: candidate seuqential patterns of all sequences
configure:
1. var localMap<itemset, support>;
   // used to locally aggregate supports.
map:
2. var data = read input data;
3. if(data.itemset is in localMap)
4.   localMap(itemset).support++;
5. else
6.   insert <itemset, 1> into localMap;
close:
7. for(each <itemset, support> pair in localMap)
8.   output <itemset, support>;

SAReducer
reduce(in_key, in_values):
1. var count = 0;
2. for(each value in in_values)
3.   count += value;
4. if(count >= minimum support)
5.   output<in_key, count>;
output: frequent sequential patterns with their supports
```

Fig. 2. Candidates Computing Job and Supports Assembling Job

a candidate sequential pattern. In lines 13 to 14, if the value is the arriving itemset of a sequence, CCReducer stores the input itemset for the generation of new candidate itemsets in the following lines. It is noted that there is only one newly arriving itemset of a specific sequence number at a timestamp.

CCReducer has to compute all combinations of items in the arriving itemset in order to generate the complete set of different sequential patterns. For example, if the incoming itemset is (ABC), all combinations for generating candidate sequential patterns are A, B, C, (AB), (AC), (BC), and (ABC). In lines 17 to 22, CCReducer first appends each combination to each candidate itemset in the cand_set summary to form new candidate sequential patterns. Then, the newly generated candidate sequential pattern is put into the result set as an output in line 19. Meanwhile, if the timestamp of the candidate sequential pattern is bigger than the start time of the current POI, the newly generated candidate itemset is put in the summary of the current timestamp for further computation at the next timestamp in lines 20 to 21. Note that the candidate itemsets whose timestamps equal to the start time are not stored. In other words, the obsolete data at the next timestamp are pruned away. In addition to the newly generated candidates, CCReducer stores each combination with the current timestamp in the summary at the current timestamp in line 23. The summary of the current timestamp will be used to compute candidate sequential patterns at the next timestamp. Finally, all candidate itemsets in the result set are output as <candidate itemset, null> pairs in lines 25 to 26. After the collection of output pairs of each CCReducer, the candidate computing job has dealt with all incoming itemsets at the current timestamp, generated candidate sequential patterns of all sequences in the current POI, and updated candidate set summaries of all sequences for the computation at the next timestamp.

3.2 Support Assembling Job

As shown in Figure 2, the support assembling job calculate supports for each candidate sequential patterns. The support assembling job reads all candidate sequential patterns from the outputs of the candidate computing job. SAMapper utilizes a local map to aggregate occurrence frequencies of different candidate sequential patterns locally in lines 2 to 6 and outputs <candidate sequential pattern, its local supports> pairs in lines 7 to 8. Pairs with the same candidate sequential pattern are sent to the same SAReducer. In lines 2 to 3 of SAReducer, SAReducer accumulates supports of the same candidate sequential pattern again and gathers the final supports. For those candidate sequential patterns whose supports are no less than the minimum support threshold, SAReducer reports them as frequent sequential patterns in the current POI in lines 4 to 5. Then, DPSP algorithm moves to the next timestamp and repeats these Map/Reduce jobs.

4 Performance Evaluation

4.1 Experimental Designs

To assess the performance of DPSP, we conduct several experiments to evaluate the performance and the effects of input parameters. DPSP is implemented in Java and runs on top of Hadoop version 0.19.1. Hadoop cluster consists of 13 nodes and each node contains 2 intel Xeon(TM) CPU 3.20GHz, 2GB RAM and 32GB SCSI hard disk. The synthetic datasets are generated the same as [7]. In our experiments, every point in the figures is the total execution time of 40 timestamps and the POI is set as 10 timestamps unless specified otherwise. The minimum support threshold is set to 0.02 and there are 10000 different items in the synthetic datasets.

4.2 Experimental Results

First, we examine the performance of DPSP with large numbers of sequences as shown in Figure 3. Note that both X-axis and Y-axis are in log scale in (a). The total execution time does not vary a lot when the number of sequences is

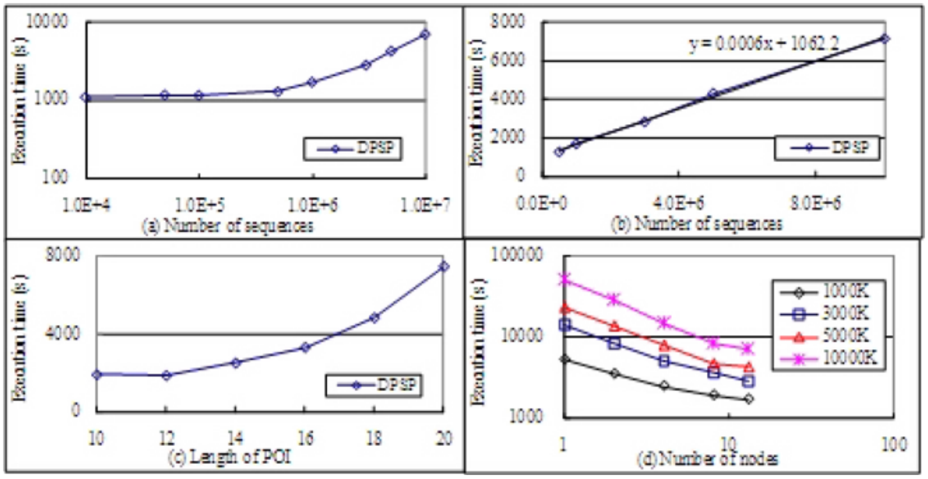


Fig. 3. Experiments

smaller than 500k. The reason is that most of the execution time comes from the overhead of Hadoop scheme such as disk I/O and communication costs. When the number of sequences is bigger than 500k, the total execution time increases linearly. We show the linear part of Figure 3(a) in more details in Figure 3(b). The linear equation of the regression line is y = 0.0005x + 1057.8, which means DPSP possesses very good scalability. Therefore, DPSP shows great practicability with large number of sequences. In the second experiment, we demonstrate the effect of increasing the length of POI. As shown in Figure 3(c), the total execution time goes up very quickly. The reason is that the number of candidate sequential patterns generated by each sequence grows exponentially as the length of POI increases. Therefore, the processing time of DPSP increases accordingly. The distributed nature of DPSP helps a little.

Finally, we show the advantages of the distributed scheme of our proposed algorithm DPSP. The datasets contain 1000k to 10000k sequences. As shown in Figure 3(d), the total execution time drops as the number of nodes increases from 1 to 8. This shows the merits of the distributed scheme. It is noted that both X-axis and Y-axis are in log scale. However, the overheads of disk I/O and message communication retard the reduction rate of the total execution time when the number of nodes equals to 13. Nevertheless, the decrease of the total execution time is remarkable. It is still worth to include more computing nodes in the cluster if we want to deal with more sequences. By utilizing Hadoop platform, it is extremely easy to extend the scale of the cluster to acquire better performance.

5 Conclusions

We proposed a distributed algorithm DPSP to address the inevitable scalability problem of the progressive sequential pattern mining. DPSP is running on top of Hadoop. We designed two Map/Reduce jobs in DPSP to efficiently compute candidate sequential patterns, update summaries of sequences, and assemble supports of candidate sequential patterns within each POI. As such, DPSP is able to report the most up-to-date sequential patterns. The experimental results show that DPSP possesses great scalability and thus increases practicability when the number of sequences become larger. In addition, by utilizing Hadoop platform, it is easy to increase the number of computing nodes in the cluster to acquire better performance.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proc. of Intl. Conf. on Data Engineering, February 1995, pp. 3–14 (1995)
2. Aseervatham, S., Osmani, A., Viennet, E.: bitspade: A lattice-based sequential pattern mining algorithm using bitmap representation. In: Proc. of Intl. Conf. on Data Mining (2006)
3. Cheng, H., Tan, P.-N., Sticklen, J., Punch, W.F.: Recommendation via query centered random walk on k-partite graph. In: Proc. of Intl. Conf. on Data Mining, pp. 457–462 (2007)
4. Chilson, J., Ng, R., Wagner, A., Zamar, R.: Parallel computation of high dimensional robust correlation and covariance matrices. In: Proc. of Intl. Conf. on Knowledge Discovery and Data Mining, August 2004, pp. 533–538 (2004)
5. Dean, J., Ghemawat, S.: Mapreduce: Simplified dataprocessing on large clusters. In: Symp. on Operating System Design and Implementation (2004)
6. Hadoop, http://hadoop.apache.org
7. Huang, J.-W., Tseng, C.-Y., Ou, J.-C., Chen, M.-S.: A general model for sequential pattern mining with a progressive database. IEEE Trans. on Knowledge and Data Engineering 20(9), 1153–1167 (2008)
8. Kargupta, H., Das, K., Liu, K.: Multi-party, privacy-preserving distributed data mining using a game theoretic framework. In: Proc. of European Conf. on Principles and Practice of Knowledge Discovery in Databases, pp. 523–531 (2007)
9. Luo, P., Xiong, H., Lu, K., Shi, Z.: Distributed classification in peer-to-peer networks. In: Proc. of Intl. Conf. on Knowledge Discovery and Data Mining, pp. 968–976 (2007)
10. Nguyen, S., Sun, X., Orlowska, M.: Improvements of incspan: Incremental mining of sequential patterns in large database. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 442–451. Springer, Heidelberg (2005)
11. Wolff, R., Bhaduri, K., Kargupta, H.: A generic local algorithm for mining data streams in large distributed systems. IEEE Trans. on Knowledge and Data Engineering 21(4), 465–478 (2009)
12. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.J.: Scan: A structural clustering algorithm for networks. In: Proc. of Intl. Conf. on Knowledge Discovery and Data Mining, pp. 824–833 (2007)

An Approach for Fast Hierarchical Agglomerative Clustering Using Graphics Processors with CUDA

S.A. Arul Shalom[1], Manoranjan Dash[1], and Minh Tue[2]

[1] School of Computer Engineering, Nanyang Technological University
50 Nanyang Avenue, Singapore
{sall0001,asmdash}@ntu.edu.sg
[2] NUS High School of Mathematics and Science
20 Clementi Avenue 1, Singapore
{h0630082}@nus.edu.sg

Abstract. Graphics Processing Units in today's desktops can well be thought of as a high performance parallel processor. Each single processor within the GPU is able to execute different tasks independently but concurrently. Such computational capabilities of the GPU are being exploited in the domain of Data mining. Two types of Hierarchical clustering algorithms are realized on GPU using CUDA. Speed gains from 15 times up to about 90 times have been realized. The challenges involved in invoking Graphical hardware for such Data mining algorithms and effects of CUDA blocks are discussed. It is interesting to note that block size of 8 is optimal for GPU with 128 internal processors.

Keywords: CUDA, Hierarchical clustering, High performance Computing, Computations using Graphics hardware, complete linkage.

1 Introduction

High performance computing on various multi-core processors remains as a challenge in the software industry. Amidst the presence and growth of CPU based parallel and distributed computing, the field of General Purpose Computation on Graphics Processing Unit (GPGPU) has shown tremendous achievements in increasing the computational speed by few folds. [1, 2, 9, 10, 11].

1.1 Computing Trends and Challenges Using GPU

The launch of NVIDIA's Compute Unified Device Architecture (CUDA) technology is a catalyst to the phenomenal growth of the application of Graphics Processing Units to various scientific and data mining related computations. The skills and techniques needed in invoking the internal parallel processors of a GPU should be viable to Data mining programmers who might not be expert Graphics Programmers. The intension of this work is to implement Hierarchical Agglomerative Clustering (HAC) algorithms using CUDA and demonstrate the speed gains.

1.2 Graphics Processors and CUDA Technology

The GPU is designed to perform computations with extreme speed. The raw computational power of GPU can be expressed and compared with that of the CPU in terms of 'Peak floating-point operations per second' and 'Memory Bandwidth'. Fig. 1 shows the growth trend of computational power of the GPU and the CPU in terms of Peak Giga Flops (GFlops). The NVIDIA 8800 GTS GPU has a Peak performance of 624 GFlops and Memory Bandwidth of 62 Giga Bytes per second whereas a 3.0GHz Dual Core Intel CPU has a Peak Floating point rate of 12 GFlops and Memory Bandwidth of 12.8 Giga Bytes per second. Such form of raw power of the Graphics hardware is thus available to be utilized for non-image processing related computations [4, 8, 10].

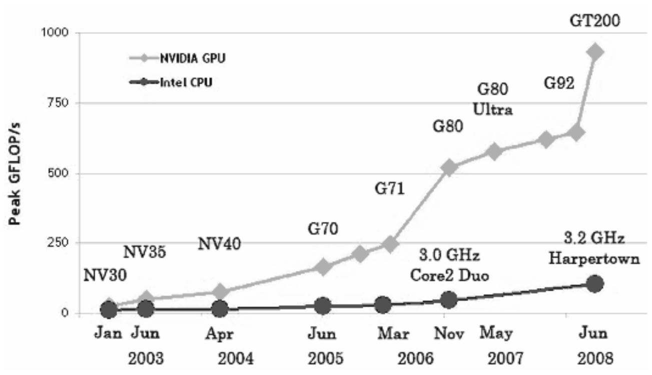


Fig. 1. Growth trend of NVIDIA GPU vs. CPU (Courtesy: NVIDIA)

2 Choice of HAC Algorithms and Implementation

2.1 HAC Algorithms

HAC is a common and important algorithm used in Data mining in the domains of micro array analysis, genome clustering, image processing and web page clustering. Hierarchical clustering seeks to build up a hierarchy of clusters. In the agglomerative approach each data vector is considered as a cluster in its own and pairs of such clusters are merged and the hierarchy moves up [3]. A measure of similarity between the clusters is required to decide which clusters should be merged. We use the Euclidean distance as the metric to determine the similarities between pair of clusters. [7, 8] where the vector norms a_i and b_i can be calculated using Equation (1), where n is the number of cluster vectors to be merged. Selection of a pair of clusters to merge depends on the linkage criteria. There are two commonly used linkage criteria and the type of HAC depends on the linkage criteria used to merge the pair of clusters.

$$\| a-b \|_2=\sqrt{\sum_{i=1}^{n}(a_i-b_i)^2} \ . \tag{1}$$

The criteria used by the complete linkage clustering and the single linkage clustering are given in Equation (2) and Equation (3) respectively.

$$Maximum\{\, dist(\, a,b\,) : a \in A, b \in B \,\}\ . \tag{2}$$

$$Minimum\{\, dist(\, a,b\,) : a \in A, b \in B \,\}\ . \tag{3}$$

The pair of clusters selected based on a criteria are merged and a cluster vector is updated. The centroid of the individual cluster vectors within the merged pair is computed to replace the original cluster vectors. The centroid C_j of k cluster vectors to be merged can be computed using Equation (4).

$$Cj = \frac{a_1 + a_2 + ... + a_k}{k} \tag{4}$$

The resultant hierarchy of clusters can be presented as a dendrogram. In this research paper, we intend to implement and analyze the results of HAC based on complete linkage and the single linkage methods. The HAC single linkage method has been previously implemented using CUDA [3]. We find that the merging of clusters was not done by computing the centroid of all the individual clusters within the pair of clusters selected for the merge and that short coming is rectified.

2.2 HAC Implementations Using CUDA

The computational steps that can be made parallel are implemented on the Graphics processor. Table 1 summarizes the functions used in the implementation of HAC complete linkage method using CUDA on GPU. This implementation architecture is common for both the single linkage with Centroids and the complete linkage methods. Understanding the CUDA architecture is vital in effectively implementing computational algorithms on the GPU and is well explained. [1, 3, 5]

Table 1. CUDA functions in HAC Complete Linkage

Computational Steps in HAC	GPU CUDA functions	Kernel in GPU?
Compute distances	*calculateDistance();*	Yes
Compute Centroid	*updateArray0();*	Yes
Update similarity half matrix	*updateArray1();* All distances from j^{th} cluster are set at *d*.	Yes
Identify maximum distance vectors	*updateArray2();*	Yes
Update similarity half matrix	*updateArray3();* i^{th} cluster distances are recalculated.	Yes
Update the i^{th} and j^{th} cluster vectors	*updateArray4();*	Yes

3 HAC Implementation Results and Discussions

The CUDA implementations of the HAC Algorithms are executed and tested on a NVIDIA GeForce 8800 GTS GPU with a memory of 512MB. The corresponding CPU implementation is run on a desktop computer with Pentium Dual Core CPU, 1.8 GHz. 1.0GB RAM on MS Windows Professional 2002. Gene expressions of Human

Mammary Epithelial Cells (HMEC) with 60000 Genes and 31 features each were used to evaluate the performance compared to the CPU implementation. This microarray dataset has been obtained from experiments conducted on HMEC. The performance of the GPU over the CPU is expressed as computational '*Speed Gain*', which is simply the ratio between the CPU computational time and the GPU computational time. The GPU computational time includes the time taken to transfer the input vectors from the CPU to the GPU and transfer cluster results back to CPU.

3.1 Determination of Optimal Block Size Based on Speed Gains

One of the parameter that affects the computational performance of GPU is the Block size. Block size in CUDA determines the number of threads to be invoked during run time. A block size of 8 invokes 8 x 8 = 64 threads during runtime which could be run in parallel. Each thread independently operates on a vector and thus exploits the parallel computing power of the GPU. Fig. 2 shows the results obtained by implementing the complete linkage HAC algorithm on the GPU with various block sizes using 5000 genes versus different dimensions. Results show that the block size of 8 is optimal for any selected number of dimensions which was used further.

3.2 Speed Gain Profile Using the Gene Expression Data Set

Fig. 3 shows the Speed Gain versus the number of Genes with 31 features for both the single linkage with Centroids and the complete linkage method. It can be noted that the single linkage method can be about 44 times faster than the CPU implementation when there are 10000 Genes to be clustered, whereas the complete linkage method reaches only about 20 times the speed of the CPU implementation above which the CPU took too long to complete, hence aborted.

3.3 Effect of Gene Size and Dimensions on Speed Gain in HAC Single Linkage

Significant resources of the GPU are utilized for the calculation of half distance matrices. For this experiment with single linkage HAC algorithm, the number of dimensions is artificially increased and the computational time taken is measured for 10000 Genes. Fig. 4 shows the effect of increase in dimensions of Genes on computational Speed Gains and the % of time taken to compute the half similarity matrices. It can be noticed that speed up to about 90 times is gained at low dimensions and it drops as the dimensions increase to about 13 to 15 times. Fig. 5 contrasts the performance with 6 and 31 dimensions.

4 Research Issues with Clustering Algorithms on CUDA

Data mining algorithms are often computationally intense and repetitive in nature which exhibit rich amounts of data parallelism. Data parallelism is a characteristic of a computational program whereby arithmetic operations can be performed on data vectors simultaneously. The inherent parallelism in the graphics hardware is invoked by CUDA features. Fig. 6 shows the CUDA architecture and the hardware resources which are used in the invocation of HAC computations on the GPU [5].

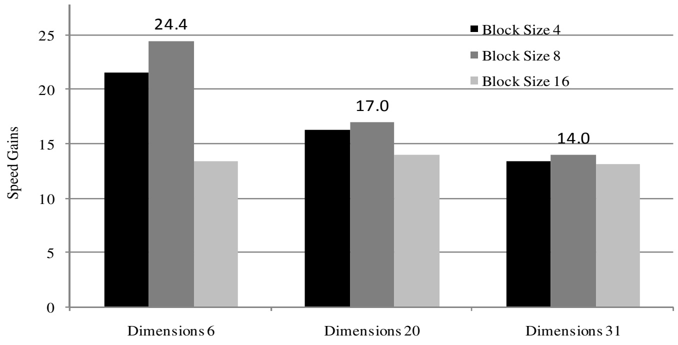


Fig. 2. HAC Complete linkage Speed Gains vs. Dimensions and CUDA Block sizes

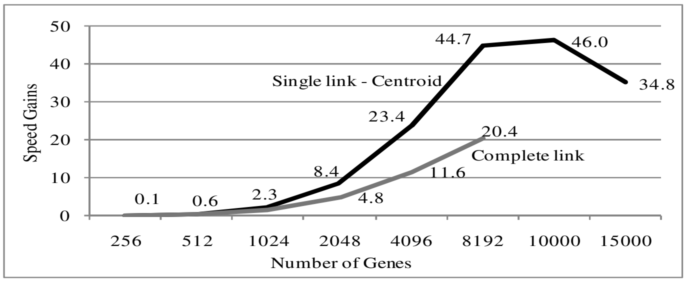


Fig. 3. GPU Performance: Single link – Centroid and Complete link vs. number of Genes

4.1 CUDA Process Block Size and Threads

Each CUDA processing block is run on one Multiprocessor (MP). In an 8800 GTS GPU there are 16 such MPs with 8 internal processors each that makes the total number of internal processors to 128. Thus while using a block size of 8, the use of 8 * 8 = 64 threads is referred to. Each thread is associated with an internal processor during runtime. Internal processors which do not belong to a block cannot be accessed by that block. So there will be a maximum of 8 execution cycles to process 64 threads while the block size is 8. If 8 such blocks can be used simultaneously, then all the 128 internal processors can be used simultaneously, thus fully harnessing the power of the GPU. This also explains why the Speed Gains with block size of 8 is high.

There is no direct control possible at block-level for the programmer and the block allocation to internal processors cannot be determined. When a grid is launched, CUDA automatically allocates blocks into the processors. There will be '$n*n/(2*k)$' threads created per block, where n is the number of observations and k is the possible number of threads per block. In Hierarchical clustering, '$n*n/2$' is the size of the half-similarity matrix, which is also the number of threads needed to simultaneously operate the entire matrix. In this HAC implementation only one block is used per grid. Hence only 1 MP is used and thus for block size 8, the number of threads invoked per block is 64. For the total number of threads generated to be invoked in the block, only

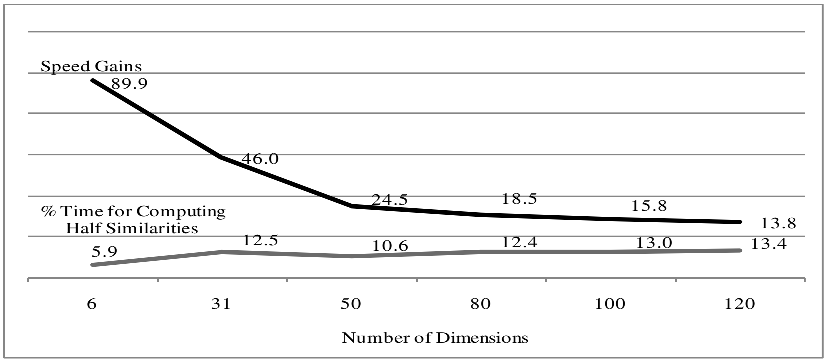


Fig. 4. HAC Single linkage method: Speed Gains with 10000 Genes vs. Dimensions and % of Time taken to compute half similarity matrices

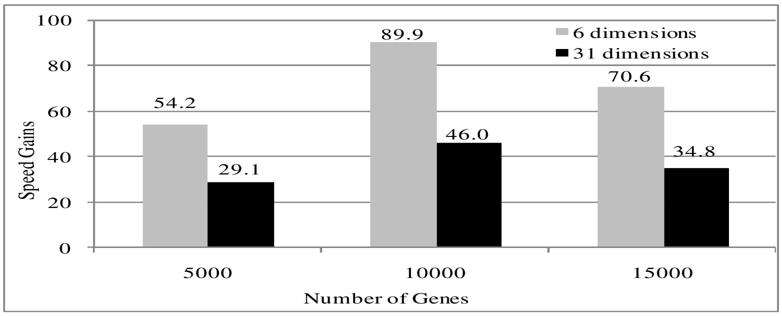


Fig. 5. HAC Single linkage Speed Gains: Gene Dimensions vs. number of Genes

'*k*number of blocks*' will be executed simultaneously. Though the total number of blocks required is '*n*n*/(2**k*)', there will be queuing while only one block is used. Within a grid, a number of blocks used will be queued up with threads and allocated to processors in a MP.

The CUDA program should use 16 or 12 or 4 blocks to fill the GPU depending on the number of internal processors in the GPU used. To be effective we need to use more blocks within the grid. When a grid is launched, the number of blocks processing is equal to '*number of MP used**8'. The number of block is designated as twice as the number of MP because computations in some of the blocks may finish earlier than the others. When a computational queue is complete, the processor will be idle and that is a waste. One way to overcome this issue is to use multiple blocks thus managing and utilizing the hardware resources of the GPU more effectively.

4.2 Analysis of Threads in CUDA for Data Parallelism

The number of threads invoked via a program is dependent on the algorithmic design. For example, for computing the vector distance of array A and array B of size n, there would be at least n threads operating on a pair of element (a_i, b_i), where $1 \le i \le n$. This

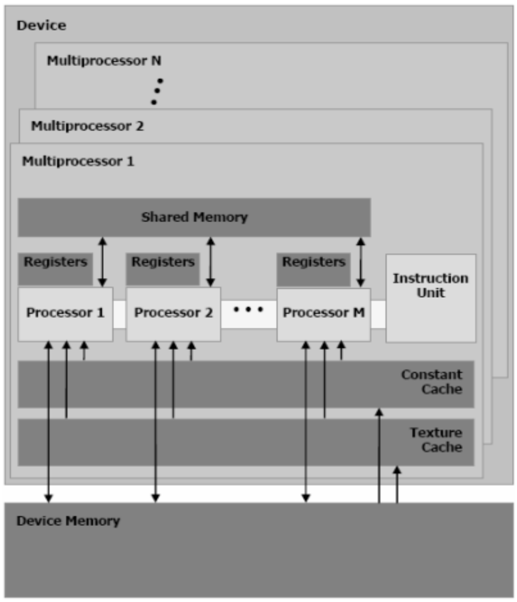


Fig. 6. GPU hardware model based on CUDA architecture (Courtesy: NVIDIA)

design theoretically would provide maximum parallelization. In the distance computations, *n* threads are arranged in a way to naturally group into blocks of similar size, satisfying the relation: '*n* = *number of threads* = (*number of blocks*) * (*number of threads per blocks*)'.

Each thread in a block is given a unique *thread-index*, in order to identify threads in different blocks. Therefore, to differentiate any two threads a *thread ID* can be conceived as follows: '*thread ID* = (*block-index, thread-index*)' where block-index is unique among any block. Blocks are organized into a grid. Thread-index and block-index may be formed of one, two or three dimensions. For the computations in HAC methods, it is found easier to conceive a one-dimensional grid [3].

5 Conclusion

We implemented single linkage centroid and the complete linkage HAC methods. Speed Gains about 15 to 90 times than the CPU have been achieved. The computational speed gain on HAC single linkage method is almost twice as obtained for the complete linkage method. This is due to the fact that the identifying maximum distance pair needs a custom developed function whereas the identification of minimum distance pair uses the built in CUDA library *(cublasIsamin)* function. The issues rising from the implementation of HAC methods using CUDA have been discussed and generalized. The optimal block size for CUDA processing on GPU with 128 internal processors should be 8. Maximum number of observations that can be currently clustered is limited by the size of distance matrix. Future plans include the use of 'Multiple Blocks' and implementing variants of HAC algorithm.

References

1. Halfhill, T.R.: Parallel Processing with CUDA. In: Nvidia's High-Performance Computing Platform Uses Massive Multithreading (2008)
2. Reuda, A., Ortega, L.: Geometric algorithms on CUDA. Journal of Virtual Reality and Broadcasting n(200n), Departamento de Informática Universidad de Jaén, Paraje Las Lagunillas s/n. 23071 Ja´en – Spain
3. Arul, S., Dash, M., Tue, M., Wilson, N.: Hierarchical Agglomerative Clustering Using Graphics Processor with Compute Unified Device Architecture. In: International Conference for Computer Applications, Singapore (2009)
4. Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A.E.: A survey of general-purpose computation on graphics hardware. In: Proc. Eurographics, Eurographics Association, Eurographics'05, State of the Art Reports STAR, August, pp. 21–51 (2005)
5. NVIDIA Corportation. CUDA Programming Guide 2.0. NVIDIA CUDA developer zone (2008), `http://developer.download.nvidia.com/compute/cuda/2_0/docs/NVIDIA_CUDA_Programming_Guide_2.0.pdf` (retrived December 12, 2008)
6. Causton, H.C., Quackenbush, J.: Microarray Gene Expression Data Analysis. Blackwell Publishing, Malden (2003)
7. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons, Inc., New York (1990)
8. Chang, D., Nathaniel, A., Dazhuo, J., Ming, O.L.: Compute pairwise euclidean distances of data points with GPUs. In: Proc. IASTED International Symposium on Computational Biology and Bioinformatics (CBB), Orlando, Florida, USA (2008)
9. Wilson, J., Dai, M., Jakupovic, E., Watson, S.: Supercomputing with toys: Harnessing the power of NVDIA 8800GTX and Playstation 3 for bioinformatics problems. In: Proc. Conference Computational Systems Bioinformatics, University of California, San Diego, USA, pp. 387–390 (2007)
10. Govindaraju, N., Raghuvanshi, R., Manocha, D.: Fast approximate stream mining of quantiles and frequencies using graphics processors. In: Proc. ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, pp. 611–622 (2005)
11. Zhang, O., Zhang, Y.: Hierarchical clustering of gene expression profiles with graphics hardware acceleration. Pattern Recognition Letters 27, 676–681 (2006)

Ontology-Based Mining of Brainwaves: A Sequence Similarity Technique for Mapping Alternative Features in Event-Related Potentials (ERP) Data

Haishan Liu[1], Gwen Frishkoff[2,3], Robert Frank[2], and Dejing Dou[1]

[1] Computer and Information Science Department, University of Oregon, University of Oregon, Eugene, OR 97403
[2] NeuroInformatics Center, University of Oregon, University of Oregon, Eugene, OR 97403
[3] Department of Psychology & Neuroscience Institute, Georgia State University, Atlanta, GA 30303

Abstract. In this paper, we present a method for identifying correspondences, or mappings, between alternative features of brainwave activity in event-related potentials (ERP) data. The goal is to simulate mapping across results from heterogeneous methods that might be used in different neuroscience research labs. The input to the mapping consists of two ERP datasets whose spatiotemporal characteristics are captured by alternative sets of features, that is, summary spatial and temporal measures capturing distinct neural patterns that are linked to concepts in a set of ERP ontologies, called NEMO (Neural ElectroMagnetic Ontologies) [3, 6]. The feature value vector of each summary metric is transformed into a point-sequence curve, and clustering is performed to extract similar subsequences (clusters) representing the neural patterns that can then be aligned across datasets. Finally, the similarity between measures is derived by calculating the similarity between corresponding point-sequence curves. Experiment results showed that the proposed approach is robust and has achieved significant improvement on precision than previous algorithms.

Keywords: Schema Matching, Sequence Similarity Search, ERP Data.

1 Introduction

Over the last two decades, neuroscience has witnessed remarkable advances in the development of methods for research on human brain function, including high-density electroencephalography (EEG) and event-related potentials (ERP). The ERP ("brainwave") method is a direct measure of neuronal activity.. ERP methods have yielded a large number of patterns that are associated with various behavioral and cognitive functions [12, 13]. Remarkably, however, there are few quantitative comparisons ("meta-analyses") of ERP data from different studies.. The inability to compare results across experiments has made it difficult to achieve a high-level synthesis and understanding of the vast majority of ERP results.

To address this problem, we have been working to design a system, called Neural ElectroMagnetic Ontologies, or NEMO [3, 6], for data sharing and integration of results across different ERP analysis methods, experiment paradigms, and research sites with the help of formal ontologies. In the present paper, we extend this prior work by introducing a method for identifying correspondences, or mappings, between alternative sets of ERP spatial and temporal measures. These alternative measures reflect different ways that ERP pattern features can be summarized. For example, one research group might use a measure of peak latency (time of maximum amplitude) to summarize the timecourse of the "P100" pattern in a visual object processing experiment [14, 15], while another group might use measures of pattern onset and offset to characterize the same data. Given that different analysis methods may yield distinct and complementary insights, it is likely that this "embarrassment of riches" in ERP analysis will persist. The challenge then becomes how to develop an automatic way to find valid correspondences between features of ERP datasets that are derived from different analysis methods.

To this end, we create simulated ERP data using a tool that we develop, called NEMOautolabel[1]. We extract alternative measures of latency and scalp topography (see Appendix in [5] for example) to simulate heterogeneities that arise when distinct measure generation techniques are adopted by two different research groups. Our goal is then to discover mappings between the alternative measures. This is essentially a schema mapping (or matching, we use them interchangeably in the present paper) problem as the alternative sets of measures are served as features in different ERP datasets. Due to the nature of the ERP data, we face several unique challenges:

1. Useful schema information is limited, since the data under study is semi-structured.
2. Language-based or linguistic schema-level matcher that makes use of name and text similarity is not suitable, since alternative features of ERP datasets often use totally different names (see experiments in Section 4 for example).
3. Values of alternative measures are numerical. Conventional instance-level matcher that handles mapping between numerical elements based on extraction of statistical characterization, such as range, mean and standard deviation, are not appropriate, since they are too rough to capture patterns that are crucial in determining the correspondence.

To address these challenges, we propose a novel method that explores sequence similarity search techniques and the NEMO ontologies, resulting in a framework for *ontology-based mining* of ERP data. Ontology-based mining has recently emerged as a new area of data mining, in which ontologies are used as formal domain knowledge to guide the data mining process in order to enhance performance and to represent the data mining result. Our method starts by transforming the vector of values of each measure into a *point-sequence curve*, and then evaluates similarities of the curves across datasets to determine the appropriate mapping across measures. The key problem then becomes to align subsequences of values in a principled way, thus enabling valid comparisons among instances of spatial and temporal measures across datasets. If the correspondence between two measures is not known a priori (as assumed in the present study), and if the values for these two measures are plotted

[1] `http://nemo.nic.uoregon.edu/wiki/NEMO_Analysis_Toolkit`

against the arbitrary instance numbers associated with the two datasets, the resulting graph will show no clear pattern and thus no correspondence between alternative measures (see Fig. 1, left frame). Our solution is to introduce structure into these (initially random) point-sequence curves by applying clustering to extract similar subsequences, which are further labeled using terms defined in the NEMO ontologies. These subsequences can then be aligned across the datasets, and correspondences between measures established using standard techniques for time-sequence similarity search (see Fig. 1, right frame). This approach exploits prior (domain) knowledge of the patterns that are commonly seen in ERP experiments of a particular type (e.g., visual perception) while asserting no prior knowledge about the measures.

The rest of this paper is organized as follows: In Section 2 we give a brief overview of prior work on schema matching with a focus on instance-level approaches, and time-sequence similarity search. In Section 3 we present the simulated ERP data design and methods for point-sequence matching. In Section 4, we present the ERP mapping results. Finally, in Section 5, we consider the assumptions and constraints of these methods and discuss future research directions, highlighting the contributions of this work to future research on schema matching and meta-analysis of ERP data.

2 Related Works and Background

2.1 Schema Matching

Our study of mapping alternative measure sets is closely related to the schema matching problem. A schema matching algorithm may use multiple matching methods or *matchers*. It generally falls into one of two categories based on if it considers instance data or only schema information. Our ontology-based mining approach should be considered as one kind of instance-level method. According to the type of instance value, various instance-based approaches have been developed in previous research. For example:

- For textual attributes, a linguistic characterization based on information retrieval techniques can be applied [18].
- For nominal attributes, evaluation of the degree of overlap of instance values is a preferred approach. Larson *et al.* [10] and Sheth *et al.* [11] discussed how relationships and entity sets could be integrated primarily based on their domain relationships: EQUAL, CONTAINS, OVERLAP, etc. Similarity of partially overlapped instance set can be also calculated based on measures such as Hamming distance and Jaccard coefficient.
- For numeric attributes, typically one can use their values to compute statistics to characterize the attributes—e.g., 'SSN' and 'PhonNo' can be distinguishable since their data patterns, such as value distributions, and averages, are different [18].

Hybrid systems that combine several approaches to determine matching often achieve better performance. For example, SemInt [16, 17] is a comprehensive matching prototype exploiting up to 15 constraint-based and 5 content-based matching criteria. Instance data is used to enhance schema-level information by providing actual value distributions, numerical averages, etc. SemInt determines a *match signature* for each attribute for either all or a selected subset of the supported criteria. Then neural networks or distance-based similarity measures over signatures can be used for determining an ordered list of match candidates.

The LSD (Learning Source Descriptions) system uses machine-learning techniques to match a new data source against a previously determined global schema [18]. It represents a composite match scheme with an automatic combination of match results. In addition to a name matcher they use several instance-level matchers (learners) that are trained during a preprocessing step. Given an initial user-supplied mapping from a data source to the global schema, the system trains multiple learners, thereby discovering characteristic instance patterns. These patterns and rules can then be applied to match other data sources to the global schema.

The iMAP [9] system can semi-automatically discover one-to-one and even complex mappings between relational database schemas. The goal is to reformulate the matching problem as a search in a match space, To perform the search effectively, iMAP uses multiple basic matchers, called searches, e.g., text, numeric, category, unit conversion, each of which addresses a particular subset of the match space.

An important limitation of the above instance-based matching methods is their inability to properly handle numerical instances in some certain domain application. They use statistical characterization extracted from the numerical instances, such as range, mean and standard deviation, to determine match. However such information is too rough to capture patterns in ERP data that are crucial in determining the correspondence. By contrast, our proposed sequence similarity search technique is specifically designed to handle attributes with numerical values for ERP data: a spatial distance measure is used to calculate the similarity between point-sequence curves representing the numerical attributes after subsequence reordering based on clustering, as described in Section 3.

2.2 Subsequence Similarity Search

We assume that similarity between point-sequence curves implies similarity between the metrics they represent. Therefore, we view the discovery of mappings between metric sets as a similarity search among two sets of point-sequence (time series) data.

Sequence similarity search has emerged as an active area of research. In general, methods for sequence similarity search belong to one of two categories [1]: 1) Whole Matching—the sequences to be compared have the same length (after interpolation or offset adjustment if necessary); and 2) Subsequence Matching—the query sequence is smaller; we look for a subsequence that best matches the query sequence.

The ERP metric mapping problem is a whole matching problem. Furthermore, we consider the cross spatial distance join [4] problem as a special case of whole matching. The spatial distance join is defined using two datasets, A and B, and a distance function L. For a given radius r, the spatial distance join computes the following set:

$$\{\langle a, b\rangle | a \in B, L(a, b) \leq r\} .$$

The term cross spatial join emphasizes that the two point sets A and B are distinct. The distance function L represents a similarity measure.

Performing the sequence similarity search task consists primarily of making the following choices: 1) a distance function *L*; 2) a method to generate cross pairs (*a*, *b*); and 3) a usage of approximations of objects as an index to the exact representation (i.e., to calculate *L*). Agrawal *et al.* [1] point out that the choice of *L* is clearly application-dependent. Although a wide spectrum of similarity measures has been proposed, a comprehensive survey by Keogh et al [7], which carried out extensive performance tests on different similarity measures, demonstrated that Euclidean distance outperformed other distance metrics. Therefore, we chose Euclidean distance as the distance function *L* in our study.

The problem of performing efficient spatial joins in relational database systems has been studied by Brinkho *et al.* [2]. They point out that spatial join is a kind of multiple-scan query where objects have to be accessed several times and therefore, execution time is generally not linear but superlinear in the number of objects. They propose to use the R-tree family to support efficient spatial queries and manage to achieve almost optimal I/O time.

For performance issues, indexing is also essential for similarity searches in sequence databases. Indexing is a technique that extracts k features from every sequence, maps them to a k-dimensional space, and then discovers how to store and search these points. This method can help alleviate the "curse of dimensionality" and to preserve spatial locality in disk pages for I/O optimization.

In the present study, we adopt a "naïve" approach that computes similarity on every cross-join pair of conjugate sequences. The cross join is performed by multiple sequential scans of the two datasets, and we do not perform indexing on the original sequences. The rationale is that scalability is not a major concern since the number of sequences (i.e., number of measures) in most ERP datasets is relatively small (<20).

3 Methods

We propose to view the feature value vector of each ERP summary metric as forming a point-sequence curve. The problem of matching discovery between metric sets can then be framed as a sequence similarity search task. To identify structured subsequences in each feature vector, we use clustering and label discovered clusters with respect to the simulated ERP patterns or "components" (e.g., *P100*, *N100*, *N3*, *MFN*, and *P300*. All of them are defined in the NEMO ontologies). By labeling the feature instances in this way, we can group them in each dataset based on their pattern labels and then align the instance groups across datasets accordingly. This step can be viewed as a subsequence reordering process. We then apply a sequence post-processing step to achieve better performance in the similarity search, leveraging the rich collection of sequence similarity search algorithms presently available. The final step is to evaluate the similarity of the structured point-sequence curves that now represent our two simulated ERP datasets as quantified by their respective measures. This evaluation is achieved by using the cross-spatial join to calculate the distance between all pairs of sequences from the two datasets. In this way, we can discover matching pairs of measures. Each of these steps is described in the following sections.

3.1 Simulated ERPs

The raw data for this study consist of 80 simulated event-related potentials (ERPs), where each ERP comprises simulated measurement data at 150 time samples and 129 channels (electrodes) for a particular subject (*n*=40) and experiment condition (*n*=2). The 40 simulated subjects are randomly divided into two datasets, *SG1* and *SG2*, each comprising 40 ERPs (20 subjects and 2 experimental conditions). Each ERP consists of a superposition of 5 latent spatiotemporal patterns that represent the scalp projections of distinct neuronal groups (dipoles). To create these patterns of neural activity, 9 dipoles are located and oriented within a 3-shell spherical model to simulate the topographies of 5 ERP components commonly seen in studies of visual word recognition. Each dipole is then assigned a 600 ms activation consistent with the temporal characteristics of its corresponding ERP. Simulated "scalp-surface" electrode locations are specified with a 129-channel montage, and a complex matrix of simulated noise is added to mimic known properties of human EEG. Because of volume conduction and the overlap of their temporal activity, the dipole activations induce a complex spatial and temporal superposition of the 5 modeled ERP patterns.

Spatiotemporal components are extracted from the two datasets, SG1 and SG2, using two techniques: temporal Principal Components Analysis (tPCA) and spatial Independent Components Analysis (sICA), two data decomposition techniques that are widely used in ERP research. Two alternative metric sets, *m1* and *m2*, are subsequently applied to the two tPCA and the two sICA derived datasets to quantify the spatiotemporal characteristics of the extracted patterns.

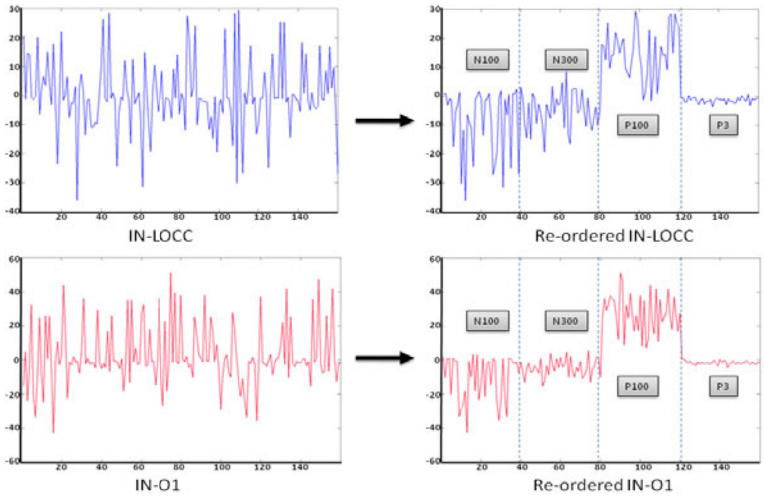


Fig. 1. (Left) *IN-LOCC* and *IN-O1* point-sequence curves prior to grouping and reordering. (Right) Labeled curves for metrics *IN-O1* and *IN-LOCC* after grouping/reordering.

3.2 Data Partitioning and Reordering

In the present study, we perform clustering on the spatial and temporal values of the two alternative sets of measures using Expectation Maximization (EM) algorithm. The resulting clusters represent candidate ERP patterns, characterized by the central tendencies of their cluster attributes (i.e., mean values for the spatial and temporal metrics). We label the resulting clusters with pattern labels defined in the NEMO ontologies (P100, N100, etc.) using rules specified by domain experts.

Following clustering and labeling, the pattern labels are used to align groups of instances across datasets, resulting in subsequence reordering. As illustrated in the right-hand graphs of Fig. 1, the point-sequence curves for metrics *IN-O1* and *IN-LOCC* (plotted using their original orderings prior to grouping/reordering on the left-hand side) are manifestly more similar after reordering subsequences in the two curves by aligning instances that belong to the same (or similar) patterns.

3.3 Sequence Post-processing

After alignment of the subsequences according to pattern labels defined in the NEMO ontology, we carry out three post-processing steps: (1) Normalization, i.e., scaling all the sequence values to unit range; (2) Smoothing, using a moving average method to reduce within cluster variance; and (3) Interpolation of curves, if the number of points in two point-sequence curves is different. Fig. 2 illustrates the results of normalization, smoothing and interpolation to the point-sequence curves of *IN-O1* and *IN-LOCC* in Fig. 1.

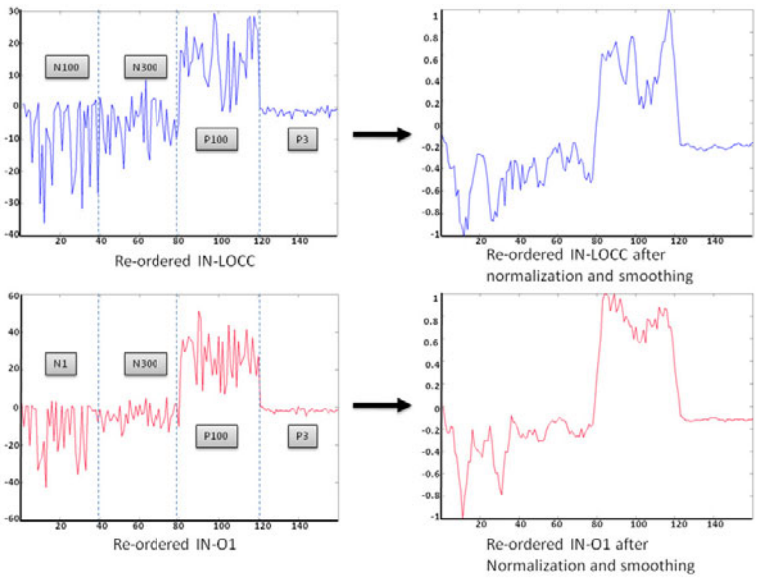


Fig. 2. After normalization, smoothing, and interpolation of point-sequence curves in Fig. 1

3.4 Sequence Similarity Search

The following heuristic assumptions are adopted in our sequence matching procedure.

First, we assume that the two datasets from which these alternative measures are extracted contain the same or similar ERP patterns. This assumption is critical, since it permits us to reorder the two point-sequence curves by aligning subsequences that are associated with the same ERP pattern labels.

Second, we assume that there exists a 1-to-1 mapping between pairs of metrics from the alternative sets of metrics. In other words, there must be no cells selected within the same column.

Table 1. Example for violation of the 1-1 mapping assumption and the solution

	IN-O1	**IN-O2**
IN-LOCC	4.08	***3.74***
IN-ROCC	4.01	***3.57***

(a)

	IN-O1	**IN-O2**
IN-LOCC	***4.08***	3.74
IN-ROCC	4.01	***3.57***

(b)

For example, Table 1(a) illustrates a scenario where the 1-to-1 mapping assumption is violated: the value in each cell is the Euclidean distance between two point-sequence curves denoted by the row and column header of the cell. If we select cells with minimum distance value in each row, we end up with two cells within the same column being selected, suggesting that both *IN-LOCC* and *IN-ROCC* are mapped to *IN-O2* in the present case. Table 1(b) illustrates the solution: cells are selected using the 1-to-1 mapping heuristic coupled with the global minimum heuristic (see below).

Finally, we assume a global minimum heuristic: we select those cells whose Euclidean distance values sum up to a minimum value.

Table 2. Solution to Table 1 using global minimum heuristic

	IN-O1	**IN-O2**
IN-LOCC	4.08	***3.74***
IN-ROCC	***4.01***	3.57

(a)

	IN-O1	**IN-O2**
IN-LOCC	***4.08***	3.74
IN-ROCC	4.01	***3.57***

(b)

For example, Table 2 shows two alternative cell selections that do not violate the 1-to-1 mapping heuristic. The global minimum heuristic requires us to favor 2(b) because $4.08 + 3.57 < 3.74 + 4.01$. The cell selections that achieve the global minimum suggest the most stable mapping result. The global minimum heuristic requires a non-greedy implementation that should take into consideration all possible selections. When the number of metrics is large, this implementation becomes more computationally challenging.

4 Results

The experiment is conducted on the simulated datasets described in Section 3.1. The test cases for the matching discovery experiment are derived as follows: each test case contains a source and target dataset that are derived respectively from one subject group (SG1 or SG2) characterized with one metric set (m1 or m2) and formulated under one decomposition method (sICA or tPCA), and from the other subject group with the alternative metric set and decomposition method. This yields 2 (subject groups) × 2 (metric sets) × 2 (decomposition method) = 8 test cases, each of which includes two different datasets, two alternative metric sets and two decomposition methods. In order to test the robustness of the proposed methods, we replicate the datasets for each test case into five copies with different random ordering of the instances, thus resulting in a total of 40 enriched test cases.

We test our method on each of these test cases. Table 3, for example, shows a distance table calculated by cross-spatial join of tPCA-derived data from SG1-m1 and SG2-m2. The highlighted cells indicate similarity pairs between two point-sequence curves representing two measures (row header and column header which meet at this cell) and are selected by using the 1-to-1 mapping and global minimum heuristics described in Section 3.4. A similarity pair represents a potential mapping discovered by our methods. For example, from this table we derive the following mappings: *IN-O1↔IN-LOCC*, *IN-O2↔IN-ROCC*, *IN-C3↔IN-LPAR*, etc. Note that the orders of the row and column header labels are such that the golden standard mapping falls along the diagonal cells. Therefore we can easily conclude that the precision of mapping in this test case is 9/13=69.2% since 4 out of 13 cells are shifted off from the diagonal.

Table 3. Cross-spatial join of data from SG1-m1 (tPCA) and SG2-m2 (tPCA)

	IN-O1	**IN-O2**	**IN-C3**	**IN-C4**	**IN-T7**	**IN-T8**	**IN-F7**	**IN-F8**	**IN-Fp1**	**IN-Fp2**	**IN-F3**	**IN-F4**	**TI-max2**
IN-LOCC	***2.76***	2.76	8.59	8.52	9.68	10.44	11.52	11.61	11.56	11.56	7.92	7.90	12.93
IN-ROCC	2.75	***2.75***	8.58	847.00	9.69	10.47	11.55	11.64	11.60	11.60	7.91	7.86	12.95
IN-LPAR	8.57	8.58	***4.13***	5.12	9.29	8.91	9.24	9.07	8.98	8.97	5.58	6.07	9.39
IN-RPAR	7.97	7.97	3.55	***4.38***	8.97	8.66	9.10	8.93	8.88	8.85	4.99	5.39	9.43
IN-LPTEM	9.32	9.34	8.54	9.23	5.00	***4.26***	5.62	5.34	5.73	5.72	7.37	7.88	11.42
IN-RPTEM	7.81	7.81	7.66	8.05	***4.18***	3.84	5.61	5.39	5.85	5.78	6.24	6.56	11.28
IN-LATEM	11.00	11.00	8.40	8.96	3.20	2.74	***2.30***	2.09	2.52	2.43	6.89	7.35	10.95
IN-RATEM	11.19	11.19	8.53	9.03	3.33	2.45	2.51	***2.08***	2.80	2.64	6.99	7.41	11.30
IN-LORB	9.58	9.58	6.00	6.48	4.23	4.50	3.58	3.63	3.35	***3.26***	4.36	4.83	10.31
IN-RORB	11.19	11.20	8.36	8.93	3.44	3.33	2.15	2.12	***2.21***	2.16	6.85	7.33	10.83
IN-LFRON	6.72	6.71	4.05	4.01	6.30	7.10	6.91	7.06	6.76	6.71	***2.74***	2.20	9.99
IN-RFRON	6.36	6.33	4.58	4.03	7.09	7.94	8.01	8.15	7.96	7.88	3.42	***3.06***	10.67
TI-max1	11.72	11.71	7.18	7.74	12.12	11.74	12.02	11.88	11.89	11.87	9.36	9.61	***8.58***

The performance of our methods among the 40 test cases is quite good. Table 4 summarizes the precision for each test case. The table consists of eight divisions, each of which illustrates the precision measures for the datasets generated by five samples of replication to one of the original eight test schemes with random instance ordering. Since the fact that the precision of mapping by making a random guess is almost zero and that the results demonstrate consistent performance on randomly ordered data, the

Table 4. Precision results for 40 test cases

(SG1, sICA, m1) *vs.* (SG2, sICA, m2)		**(SG1, tPCA, m1) *vs.* (SG2, tPCA, m2)**		**(SG1, sICA, m1) *vs.* (SG2, tPCA, m2)**		**(SG1, tPCA, m1) *vs.* (SG2, sICA, m2)**	
Input	Precision	Input	Precision	Input	Precision	Input	Precision
Sample 1	13/13	Sample 1	9/13	Sample 1	13/13	Sample 1	5/13
Sample 2	13/13	Sample 2	9/13	Sample 2	13/13	Sample 2	5/13
Sample 3	13/13	Sample 3	9/13	Sample 3	13/13	Sample 3	5/13
Sample 4	13/13	Sample 4	9/13	Sample 4	13/13	Sample 4	5/13
Sample 5	13/13	Sample 5	9/13	Sample 5	13/13	Sample 5	5/13
(SG2, sICA m1) *vs.* (SG1, sICA, m2)		**(SG2, tPCA, m1) *vs.* (SG1, tPCA m2)**		**(SG2, sICA, m1) *vs.* (SG1, tPCA, m2)**		**(SG2, tPCA, m1) *vs.* (SG1, sICA, m2)**	
Input	Precision	Input	Precision	Input	Precision	Input	Precision
Sample 1	9/13	Sample 1	9/13	Sample 1	5/13	Sample 1	7/13
Sample 2	9/13	Sample 2	9/13	Sample 2	8/13	Sample 2	7/13
Sample 3	9/13	Sample 3	9/13	Sample 3	5/13	Sample 3	7/13
Sample 4	9/13	Sample 4	9/13	Sample 4	5/13	Sample 4	7/13
Sample 5	9/13	Sample 5	9/13	Sample 5	5/13	Sample 5	7/13

precision of our method appears markedly robust. Combining the mapping results in the 40 test cases into an ensemble model by a majority vote of each individual mapping, we obtain the ensemble mapping result. The overall precision is 11/13=84.6%.

We compare the performance our algorithm with SemInt [16, 17] as the baseline. Since the data contains only numerical instances, SemInt extracts from each feature value vector 5 discriminators, namely, MIN, MAX, Average, Coefficient of variance, and Standard Deviation. The feature value vector is then projected to a *match signature* characterized by these discriminators. A neural network is trained based on datasets from the 40 test cases with one metric set and tested on the rest datasets with the alternative metric set to determine the match. The result shows 19.23% precision. As we point out in Section 1, the reason why our algorithm significantly outperforms SemInt is that we are able to systematically exploit prior knowledge about patterns in ERP data that is crucial to determine the matching.

5 Conclusion and Future Work

In this paper, we describe a method for identifying correspondences, or mappings, between alternative sets of ERP measures that might be used by different ERP research labs to characterize patterns of brain electrical activity. The contributions of this work include the following:

- Use of an ontology to assign meaningful labels to ERP patterns (clusters) and thereby impose structure that is used to align alternative metrics across datasets;
- Application of sequence similarity search in discovering mappings across alternative metrics;
- Extension of the instance-level approach in schema matching, especially to handle numerical values; and
- Articulation of a global minimum heuristic in selecting 'similarity pairs' from the distance table. This heuristic proved to be useful and empirically robust in our experiment.

Mappings between alternative spatial and temporal metrics can be used to link different representations of ERP data and thus to support representations of ERP results with the help of formal ontologies [6]. In this way, our work is closely related to schema/ontology matching, which has been an active research field for a number of years [8, 18]. In the course of developing and testing these methods, we have collected a corpus of real data from different experiments [5] and have observed a large number of different kinds of heterogeneities. The presence of these heterogeneities suggests that a method for identifying mappings between features or metrics across two datasets may have widespread applications for ontology-based integration, beyond the specific applications discussed in the present study.

Following we summarize some basic assumptions and limitations of the current study and then discuss some possible directions for future work.

The proposed method assumes some domain-specific knowledge, as well as certain features of the input data. First, the source and target datasets are assumed to contain the same or similar ERP patterns. If the two datasets contain dissimilar patterns, there will be few instances that can be aligned according to common pattern labels, resulting in a poor sequence similarity search result. Second, there is assumed to be a 1-to-1 mapping between alternative data metrics. This assumption may be violated in many real-world cases. For example, in ERP data, the temporal metrics TI-begin and TI-end together capture the same information as the metric TI-duration. Our method will need to be modified in the future to handle these more complex mappings.

Other challenges include the scalability of calculations for the global minimum in the distance table, which is essentially an NP-hard problem. It could be remedied by proper implementation such as dynamic programming, but remains computationally intractable when the number of metrics is very large. Future work will seek to find an appropriate approximation method that balances the interest in accuracy and scalability. In addition, the simulated ERP data used in the present study were carefully designed to mimic many, but not all, features of real ERP datasets. In particular, we minimized variability in latency and spatial distribution of patterns across the different ERPs so that the data decomposition and clustering of patterns would remain tractable and relatively straightforward to interpret. In future work, we plan to carry out more substantial tests on genuine ERP datasets, such as those that have been collected, analyzed, and stored in our NEMO ERP ontology database [20].

References

1. Agrawal, R., Faloutsos, C., Swami, A.N.: Efficient similarity search in sequence databases. In: Lomet, D.B. (ed.) FODO 1993. LNCS, vol. 730, pp. 69–84. Springer, Heidelberg (1993)
2. Brinkhoff, T., Kriegel, H.-P., Seeger, B.: Efficient processing of spatial joins using r-trees. In: SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data, pp. 237–246. ACM, New York (1993)
3. Dou, D., Frishkoff, G., Rong, J., Frank, R., Malony, A., Tucker, D.: Development of NeuroElectroMagnetic Ontologies (NEMO): A Framework for Mining Brain Wave Ontologies. In: Proceedings of the 13th ACM International Conference on Knowledge Discovery and Data Mining (KDD 2007), pp. 270–279 (2007)

4. Faloutsos, C., Seeger, B., Traina, A., Traina Jr., C.: Spatial join selectivity using power laws. In: SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data, pp. 177–188. ACM, New York (2000)
5. Frishkoff, G.A., Frank, R.M., Rong, J., Dou, D., Dien, J., Halderman, L.K.: A Framework to Support Automated Classification and Labeling of Brain Electromagnetic Patterns. Computational Intelligence and Neuroscience (CIN), Special Issue, EEG/MEG Analysis and Signal Processing 2007 13 (2007)
6. Frishkoff, G., Le Pendu, P., Frank, R., Liuand, H., Dou, D.: Development of Neural Electromagnetic Ontologies (NEMO): Ontology-based Tools for Representation and Integration of Event-related Brain Potentials. In: Proceedings of the International Conference on Biomedical Ontology, ICBO 2009 (2009)
7. Keogh, E., Kasetty, S.: On the need for time series data mining benchmarks: A survey and empirical demonstration. Data Min. Knowl. Discov. 7(4), 349–371 (2003)
8. Wache, H., Vogele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hubner, S.: Ontology-based integration of information: A survey of existing approaches. In: IJCAI-01 Workshop: Ontologies and Information Sharing, pp. 108–117 (2001)
9. Dhamankar, R., Lee, Y., Doan, A., Halevy, A.Y., Domingos, P.: iMAP: Discovering Complex Mappings between Database Schemas. In: Proceedings of the ACM Conference on Management of Data, pp. 383–394 (2004)
10. Larson, J., Navathe, S., Elmasri, R.: A theory of attributed equivalence in databases with application to schema integration. IEEE Transactions on Software Engineering 15(4), 449–463 (1989)
11. Sheth, A., Larson, J., Cornelio, A., Navathe, S.: A tool for integrating conceptual schemas and user views. In: Proc. 4th International Conference on Data Engineering (ICDE), Los Angeles, CA, US, pp. 176–183 (1988)
12. Gratton, G., Coles, M.G.H., Donchin, E.: A procedure for using multi-electrode information in the analysis of components of the event-related potential: Vector filter. Psychophysiology 26(2), 222–232 (1989)
13. Spencer, K.M., Dien, J., Donchin, E.: A componential analysis of the ERP elicited by novel events using a dense electrode array. Psychophysiology 36, 409–414 (1999)
14. Donchin, E., Heffley, E.: Multivariate analysis of event-related potential data: a tutorial review. In: Otto, D. (ed.) Multidisciplinary Perspectives in Event-Related Brain Potential Research, pp. 555–572. U.S. Government Printing Office, Washington (1978)
15. Picton, T.W., Bentin, S., Berg, P., et al.: Guidelines for using human event-related potentials to study cognition: recording standards and publication criteria. Psychophysiology 37(2), 127–152 (2000)
16. Li, W., Clifton, C.: Semantic integration in heterogeneous databases using neural networks. In: Proc. 20th Intl. Conf. on Very Large Data Bases, pp. 1–12 (1994)
17. Li, W., Clifton, C.: SemInt: a tool for identifying attribute correspondences in heterogeneous databases using neural network. Data Knowl. Eng. 33(1), 49–84 (2000)
18. Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. The VLDB Journal 10(4), 334–350 (2001)
19. Doan, A.H., Domingos, P., Halevy, A.: Reconciling schemas of disparate data sources: a machine-learning approach. In: Proc ACM SIGMOD Conf., pp. 509–520 (2001)
20. LePendu, P., Dou, D., Frishkoff, G., Rong, J.: Ontology Database: A New Method for Semantic Modeling and an Application to Brainwave Data. In: Ludäscher, B., Mamoulis, N. (eds.) SSDBM 2008. LNCS, vol. 5069, pp. 313–330. Springer, Heidelberg (2008)

Combining Support Vector Machines and the *t*-statistic for Gene Selection in DNA Microarray Data Analysis

Tao Yang[1], Vojislave Kecman[2], Longbing Cao[1] and Chengqi Zhang[1]

[1] Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia
[2] Department of Computer Science, Virginia Commonwealth University, Richmond, VA, USA

Abstract. This paper proposes a new gene selection (or feature selection) method for DNA microarray data analysis. In the method, the *t*-statistic and support vector machines are combined efficiently. The resulting gene selection method uses both the data intrinsic information and learning algorithm performance to measure the relevance of a gene in a DNA microarray. We explain why and how the proposed method works well. The experimental results on two benchmarking microarray data sets show that the proposed method is competitive with previous methods. The proposed method can also be used for other feature selection problems.

1 Introduction

The advent of DNA microarray technology, such as the cDNA arrays and the high density oligonucleotide chips, has revolutionized the field of molecular biology in recent years. This new technology allows scientists to study thousands of genes simultaneously in a single experiment. This is a significant improvement because in the past, only several specific genes in an organism could be investigated at a time.

While the revolution generates much hope, the large amount of data obtained from microarray experiments, along with the structures of the resulting data sets, also challenges the conventional ways of analysis and modeling. One particular obstacle for analyzing a microarray data set is that often the number of genes is much greater than the number of samples; typically, the number of samples is less than a hundred, while the number of genes is usually in the thousands. In this regard, modern machine learning techniques provide a valuable toolkit for gaining insights into such data sets and extracting useful information from them.

Out of a large number of genes that exist in a microarray data set, it is often the case that most of them are irrelevant for the diagnosis of a particular disease, say, cancer, and hence are redundant. It is well-known that the performance of a modeling procedure can be significantly degraded, when many redundant genes are included in the training. Finding relevant genes can not only improve the accuracy of the resultant classifier for diagnosis purposes, but can also narrow down the potential set of cancerous genes and help gain important discipline knowledge.

Several methods for gene selection are available in the literature. One state-of-the-art technique is the method of Support Vector Machine Recursive Feature Elimination (SVM-RFE), proposed in [1]. The ranking criterion of SVM-RFE is constructed not by the intrinsic property of the data, but by the feedback from the support vector machine (SVM) classifiers. Specifically, the magnitudes of weights found by linear SVMs are used to rank the genes. At each iteration of the algorithm, a linear SVM is fitted to the training data with the remaining genes, and one or several genes are eliminated for their least significance in terms of the ranking criterion.

In this paper, we propose an alternative SVM-based method for gene selection, and call it TSVM-RFE. In particular, we consider selecting genes by combining the classical t-statistic and the modern SVM-RFE method. By taking care of the problems that may exist in either the univariate or multivariate worlds, TSVM-RFE is more robust to noisy genes than SVM-RFE and other methods, as confirmed by simulation studies.

2 Methods

In this section, we describe our gene selection method, TSVM-RFE, and illustrate its strengths, as well as giving a brief introduction to the support vector machines and t-statistic as needed by our method.

2.1 Data

The results from the microarray experiments can be represented by a matrix of expression levels. For microarray experiments having n tissue samples and p genes, the results can be represented by a $p \times n$ matrix X. In this paper, we shall focus on the classification problems with two classes, labeled by 1 and 2, respectively, and let n_k denote the sample size for class k; i.e., $n_1 + n_2 = n$. The response variable y_j, $j = 1, \ldots, n$, takes on the values of $+1$ or -1 for the two classes, respectively. For gene i, we use x_{ki} to denote the vector of values on the ith row of X that belong the class $k \in \{1, 2\}$. The mean of the values in x_{ki} is denoted by $\bar{x}_{ki}$, and the sample standard deviation by s_{ki}.

2.2 Support Vector Machines

The objective of SVMs is to find a classifier with the largest margin between the observations belonging to two different classes, while minimizing the training error. Here, the principle is that the classifier with the maximal margin is more likely to have a better generalization ability. A remarkable feature of SVMs is that the classifier is determined by only a few training samples, known as "support vectors". These vectors are borderline samples, in the sense that they are closest to the decision boundary or simply lie on the margin.

In the studies below, we shall simply use the linear support vector machines, following [1], [3]. From these and other studies, linear SVMs appear to work reasonably well for the purposes of gene selection; for nonlinear support vector machines, we refer the reader to [4]. In order to select genes, the method of support vector machine recursive feature elimination for gene selection uses the absolute weight value $|w_i|$ given in the vector of parameters $\mathbf{w}$ to rank genes.

2.3 The t-statistic

The t-statistic measures the separability between classes using a standardized distance for a single gene, which gives a relevance score for each gene. The ranking criterion is given as

$$t_i = \frac{\bar{x}_{1i} - \bar{x}_{2i}}{\sqrt{s_{1i}^2/n_1 + s_{2i}^2/n_2}}, \tag{1}$$

where $\bar{x}_{ki}$, s_{ki} and n_k are defined in Section 2.1.

2.4 TSVM-RFE

The basic idea of TSVM-RFE is to combine the t-statistic and SVM-RFE. The recursive feature elimination (RFE) algorithm [1] is used as the search engine of TSVM-RFE. In order to combine two different gene selection criteria, each criterion is transformed into a comparable scale. In particular, denoting by v the statistic used by a ranking criterion, the linear transformation

$$\sigma(v) = \frac{v - \min(v)}{\max(v) - \min(v)}, \tag{2}$$

is employed. Since $\sigma(\min(v)) = 0$ and $\sigma(\max(v)) = 1$, the range of $\sigma(v)$ is $[0, 1]$ for the training data. It is also possible to use other transformations, such as the sigmoid function or the probability function of a distribution, say, Gaussian.

From (2), the ranking statistic used for TSVM-RFE is

$$r_i = \alpha\sigma(|w_i|) + (1 - \alpha)\sigma(|t_i|), \quad 0 \le \alpha \le 1 \tag{3}$$

where w_i the weight found by linear SVMs and t_i the t-statistic. When $\alpha = 1$, TSVM-RFE is equivalent to SVM-RFE; when $\alpha = 0$, TSVM-RFE is equivalent to using the t-statistic.

To use (3), one problem remains to be solved, i.e., a value for α needs to be provided. For this, we use the 10-fold cross-validation. Specifically, a grid of α_i values are tested on the training data by 10-fold cross-validation, and the one that gives the lowest cross-validation error is deemed as "optimal". In our observation, using 11 equally spaced points for α between 0 and 1 ($\alpha = 0, 0.1, \ldots, 1$) seems enough. It is also possible to choose a finer grid, at a higher computational cost. When more than one α value produces the same cross-validation error, we use the smallest of them if $\alpha = 0$ gives a smaller cross-validation error than $\alpha = 1$; otherwise, we use the largest. In other words, the weight in this case is determined in such a way that TSVM-RFE is as close as possible to the better of the two individual methods.

2.5 An Illustration

The motivation for TSVM-RFE is to overcome the weaknesses of each individual criterion. The t-statistic is a great criterion in measuring the class separability for each individual gene. However, it can only summarize at most the patterns that exist in the univariate world. Those multivariate patterns that are common in microarray data, such as correlation among genes, may never be represented by it. By contrast, SVM-RFE is expected to capture multivariate patterns well due to its foundation in the maximal margin principle, and could outperform the t-statistic for a number of data sets. However, since support vector machines are prone to overfitting when there exist a large number of noisy genes, the t-statistic can have an advantage in such cases. It is typical in practice that both noisy genes and multivariate patterns exist in microarray data. Hence, a criterion that combines the information provided by both the t-statistic and SVM-RFE is likely to preform reasonably well: at least as well as the better of the two individuals.

The above consideration is illustrated in the following using two simple examples. As shown in the left panel of Figure 1, the two classes of a two-dimensional data set are completely linearly separable. Here, according to SVM-RFE (using $C = 1$), x_2 is more relevant than x_1, because $|w_1| < |w_2|$. From the t-statistic, x_1 is more relevant than x_2, because $|t_1| > |t_2|$. In this example, gene selection based on the t-statistic appears more reasonable than SVM-RFE. This is because statistically speaking, the variation of x_2 for separating the two classes is large, while x_1 has none. This is a situation where the maximal margin principle fails to work well.

The second example is shown in the right panel of Figure 1. In this example, the data are two-dimensional and x_1 and x_2 are positively correlated. The two classes are also linearly separable. Here, the magnitude of the t-statistic for the two features are very different: $|t_1| = 0$ and $|t_2| = 3.098$. Due to the t-statistic, x_2 is relevant and x_1 irrelevant. From SVM-RFE ($C = 1$), however, x_1 and x_2 are equally relevant, because $|w_1| = |w_2| = 0.500$. Since in this example x_1 and x_2 appear to be equally important for identifying the pattern, the t-statistic fails to select all relevant features while SVM-RFE performs well. This is a situation where a multivariate pattern exists and the support vector machines work well, but not the t-statistic.

Admittedly, the above two examples are rather crude, but they demonstrate the difficulties that, if used individually, the support vector machines and t-statistic may have for gene selection. It is not uncommon for microarray data that genes are correlated and a large number of them are irrelevant. Given this, our combined criterion is expected to perform better than each of the two individual methods; in the worst scenario, it is simply equivalent to the better one.

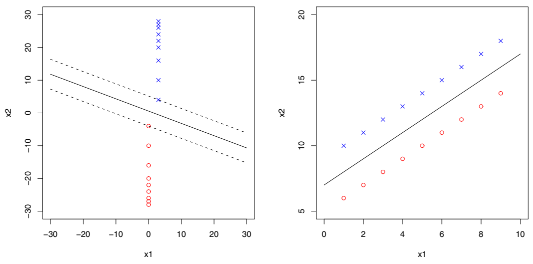


Fig. 1. Examples show the gene selection method using either the t-statistic or SVM-RFE may not be reliable. In the left panel, the gene selection method using t-statistic outperforms SVM-RFE. In the right panel, SVM-RFE outperforms the gene selection method using the t-statistic.

3 Experiments

Experiments were conducted to compare different gene selection methods and the results are given in this section. Two real data sets that are publicly available were used. A few points need to be clarified here. First, as part of pre-processing, we follow standardize each sample to mean zero and standard deviation one so as to treat each sample with an equal weight and thus to reduce array effects. Second, we follow [1] to select a fixed number of genes in the model *a priori*. In our experiments for the real data, the number of the genes retained are $10, 20, \ldots, 70$ for each method. Third, we use external cross-validation errors for comparison to avoid selection bias. Internal cross-validation errors are subject to selection bias, which are typically too optimistic [5]. Fourth, the SVM-RFE algorithm due to [3] was adopted. Fifth, a support vector machine classifier is constructed for each method after the genes are selected to assess its classification accuracy.

In the experiments, three gene selection methods, the t-statistic, SVM-RFE, and TSVM-RFE, are used to select genes. Classifiers based on linear SVM are then built using all training data, and subsequently examined using the test data.

3.1 Leukemia Data

The acute leukemia data consists of 72 samples and 7129 genes. They were obtained from Affymetrix oligonucleotide arrays. There are two types of leukemia: ALL (acute lymphocytic leukemia) and AML (acute mylogenous leukemia). We follow the procedure used in [5] to split the leukemia data set into a training set of size 38 and a test set of size 34 by sampling without replacement from all the samples, while ensuring that the training set has 25 ALL and 13 AML and the test set has 22 ALL and 12 AML. Different gene selection methods combined with linear SVMs are only applied to the training set, and then the methods are used on the test set to estimate their accuracies. Twenty such random partitions were carried out. Note that many proposed methods in the literature use a test set with size 34 only, whereas the testing procedure used here is equivalent to use a test set with 680 samples, so it is much more reliable than using the independent test set of size 34 only. In our observation, $C = 1$ is a reasonable value for the penalty parameter of SVMs in this data set.

The results for the leukemia data are summarized in Figure 2. TSVM-RFE gives the smallest minimal error of 3.68%, and strictly smaller errors than SVM-RFE and the t-statistic-based method for $30, 40, \ldots, 70$ genes. The minimal test error obtained by TSVM-RFE is also smaller than the test errors obtained by using several other methods for this data set in the literature: the minimal test error 5.00% from SVM-RFE, obtained based on fifty similar random partitions [5]; the test error 7.00% from the nearest shrunken centroid method [7]; and the minimal test error 6.00% using soft-thresholding combined with kNN classifier obtained in [7]. Interestingly, all three methods give the lowest error when 60 genes are used. This provides a reasonable suggestion for the number of relevant genes that should be used for the leukemia data.

3.2 Colon Data

The colon cancer data consists of 62 samples and 2000 genes. They were also obtained from Affymetrix oligonucleotide arrays. The task is to distinguish between the normal and tumor samples. There are 22 normal samples and 40 tumor samples in the given data. We follow the procedure used in [6] to randomly split the colon data set into a training and test set by sampling without replacement from all the samples, while ensuring that the training set has 15 normal and 27 tumor samples and the test set has 7 normal samples and 13 tumor samples. Different methods are only applied to the training set, and the test set is used to estimate the classification accuracy. Twenty such random partitions were carried out. Note that it was suggested that there were some wrongly labeled data in the training data set [5]. We follow [3] and use $C = 0.01$ for this data set.

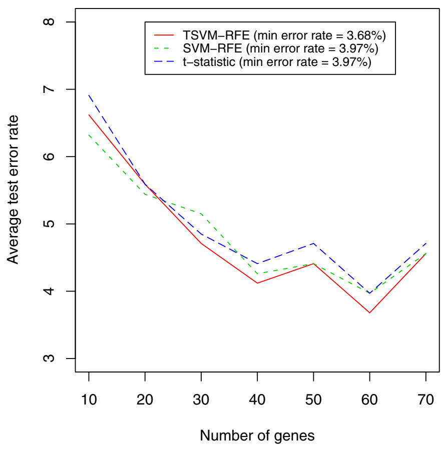


Fig. 2. Misclassification rates for the leukemia data

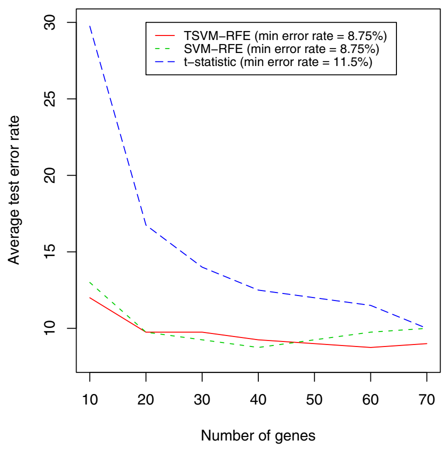


Fig. 3. Misclassification rates for the colon data

The results for the colon data are summarized in Figure 3. TSVM-RFE and SVM-RFE give the same minimal test error of 8.75%, and their performance is similar in this data set. In this data set, TSVM-RFE is always better than the method based on the *t*-statistic alone. The minimal test error obtained by TSVM-RFE here is also smaller than the test errors obtained by several other methods: the leave-one-out cross-validation error 9.68% obtained in [2], using correlation metric combined with SVMs; the minimal test error 17.50% from SVM-RFE obtained based on fifty similar random partitions [5]; the minimal jackknife error 12.50% obtained in [6] using weighted penalized partial least squares method; the minimal test error 11.16% from SVM-RFE with various values of C [3]; the test error 18.00% from the nearest shrunken centroid method [7]; the minimal test error 13.00% obtained in [7], using Wilcoxon statistic combined with kNN classifier; and the leave-one-out cross-validation error 8.90% obtained in [8], using the top scoring pair method.

4 Conclusions

We have proposed a new gene selection method, TSVM-RFE, for gene selection and classification. The criterion of TSVM-RFE combines the *t*-statistic and SVM-RFE, due to the consideration that the *t*-statistic only summarizes well the information in the univariate world and that SVM-RFE distinguishes the multivariate patterns better but is sensitive to noisy genes. We have chosen a linear transformation so that individual criteria are combined on a comparable scale, and the weight for each individual criterion is determined via cross-validation.

The proposed method was compared based on experiments with SVM-RFE and the *t*-statistic, using two practical data sets. The method presented in this paper gives competitive, if not better, results, compared to the other two. The improvement of the method proposed here upon the better-known SVM-RFE method is significant. The method presented here seems to be rather suitable for microarray data analysis, where it is likely that a large number of irrelevant genes exist and the signal-to-noise ratio is fairly low. Experiments have shown that TSVM-RFE is better than both SVM-RFE and the t-statistic, in terms of reducing misclassification errors and lowering false discovery rates. It helps to identify more accurately the truly cancerous genes.

References

1. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine Learning 46, 389–422 (2002)
2. Furey, T.S., Cristianini, N., Duffy, N., Bednarski, D.W., Schummer, M., Haussler, D.: Support vector machine classification and validation of cancer tissue samples using microarray expression data. Bioinformatics 16, 906–914 (2000)
3. Huang, T.M., Kecman, V.: Gene extraction for cancer diagnosis by support vector machines - an improvement. Artif. Intell. Med. 35, 185–194 (2005)
4. Huang, T.M., Kecman, V., Kopriva, I.: Kernel Based Algorithms for Mining Huge Data Sets, Supervised, Semi-supervised, and Unsupervised Learning. Springer, Heidelberg (2006)
5. Ambroise, C., McLachlan, G.J.: Selection bias in gene extraction on the basis of microarray gene-expression data. Proc. Natl. Acad. Sci. USA 99, 6562–6566 (2002)
6. Huang, X., Pan, W.: Linear regression and two-class classification with gene expression data. Bioinformatics 19, 2072–2078 (2003)
7. Lee, J.W., Lee, J.B., Park, M., Song, S.H.: An extensive comparison of recent classification tools applied to microarray data. Computational Statistics and Data Analysis 48, 869–885 (2005)
8. Tan, A.C., Naiman, D.Q., Xu, L., Winslow, R.L., Geman, D.: Simple decision rules for classifying human cancers from gene expression profiles. Bioinformatics 21, 3896–3904 (2005)

Satrap: Data and Network Heterogeneity Aware P2P Data-Mining[⋆]

Hock Hee Ang, Vivekanand Gopalkrishnan, Anwitaman Datta,
Wee Keong Ng, and Steven C.H. Hoi

Nanyang Technological University, Singapore

Abstract. Distributed classification aims to build an accurate classifier by learning from distributed data while reducing computation and communication cost. A P2P network where numerous users come together to share resources like data content, bandwidth, storage space and CPU resources is an excellent platform for distributed classification. However, two important aspects of the learning environment have often been overlooked by other works, viz., 1) location of the peers which results in variable communication cost and 2) heterogeneity of the peers' data which can help reduce redundant communication. In this paper, we examine the properties of network and data heterogeneity and propose a simple yet efficient P2P classification approach that minimizes expensive inter-region communication while achieving good generalization performance. Experimental results demonstrate the feasibility and effectiveness of the proposed solution.

keywords: Distributed classification, P2P network, cascade SVM.

1 Introduction

P2P networks contain large amounts of data naturally distributed among arbitrarily connected peers. In order to build an accurate global model, peers collaboratively learn [1,2,3,4] by sharing their local data or models with each other. Though recent efforts aim to reduce this communication cost compromise, none of them take into account heterogeneity in either the network or the data.

In order to build a global model representative of the entire data in the P2P network, only dissimilar data (from different data subspaces) need to be shared. While sharing similar data (from the same data subspace) adds no value to the global model, it only adds to the communication cost which can be prohibitive if the data were from geographically distant peers.

In this paper, we address the problem of learning in a P2P network where data are naturally distributed among the massive number of peers in the network. In addition, the location of these peers span across a large geographical area where *distant peers incur higher communication cost* when they try to communicate. Moreover, there is a possibility that the *data of different peers overlap in the*

[⋆] This work is partly supported by HP Labs Innovation Research Program 2008 grant.

problem space. An approach that simply exchanges data of all peers will incur a high communication cost in order to achieve high accuracy. On the other hand, an approach that does not exchange data will achieve low prediction accuracy in order to save communication costs. Hence, the objective would be to achieve the best global accuracy-to-communication cost ratio.

In this paper, we describe a data and network heterogeneity aware adaptive mechanism for peer-to-peer data-mining and study the relationship between the training problem space and classification accuracy. Our proposed approach, Satrap,

- achieves the best accuracy-to-communication cost ratio given that data exchange is performed to improve global accuracy.
- allows users to control the trade-off between accuracy and communication cost with the user-specified parameters.
- is insensitive to the degree of overlapping data among peers.
- minimizes communication cost, as the overlapping data among different regions increase.
- is simple, thus making it practical for easy implementation and deployment.

2 Background and Related Work

A P2P network consists of a large number of interconnected heterogeneous peers, where each peer holds a set of training data instances. The purpose of classification in P2P networks is to *effectively* learn a classification model from the training data of all peers, in order to accurately predict the class label of unlabeled data instances.

Existing P2P classification approaches typically either perform local [5] or distributed [1,2,4] learning. Local learning performs training within each peer without incurring any communication between peers during the training phase. Luo *et al.* [5] proposed building local classifiers using Ivotes [6] and performed prediction using a communication-optimal distributed voting protocol. Unlike training, the prediction process requires the propagation of unseen data to most, if not all peers. This incurs a huge communication cost if predictions are frequent. On the contrary, instead of propagating test instances, the approach proposed by Siersdorfer and Sizov [4] propagates the linear SVM models built from local data to neighboring peers. Predictions are performed only on the collected models, which incur no communication cost.

Distributed learning approaches not only build models from the local training data, but also collaboratively learn from other peers. As a trade-off to the communication cost incurred during training, the cost of prediction can be significantly reduced. In a recent work, Bhaduri *et al.* [2] proposed an efficient approach to construct a decision tree in the P2P network. Over time, the induced decisions of all peers converge, and as the approach is based on distributed majority voting protocol, it incurs a lower communication cost compared to broadcast based approaches.

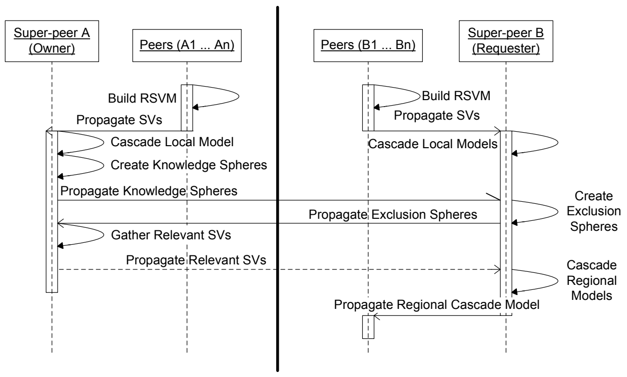


Fig. 1. Sequence diagram of Satrap (among two clusters of peers)

To reduce communication cost and improve classification accuracy, Ang *et al.* [1] proposed to cascade the local RSVM models of all peers (AllCascade). RSVM was chosen as it significantly reduces the size of the local model. However, AllCascade requires massive propagation of the local models and the cascading computation is repeated in all peers, wasting resources due to duplications.

3 Approach

Figure 1 depicts the process for constructing a global classification model in Satrap between two clusters of peers (i.e., communications are performed in a pairwise manner between different regions). Rather than flooding the entire network with models (as in AllCascade), here each peer builds an RSVM on its local data, and propagates it only within its own geographic region. This is feasible as intra-region communication is inexpensive.

Then one distinguished peer is elected from each region as the super-peer, which combines (and compresses) the models received into a regional model, and transfers them to other regions through their respective super-peers. These super-peers serve as a single point of communication between regions[1], thus reducing expensive inter-regional communication. However, note that the use of super-peers doesn't lead to a single point of failure, since if one fails, another peer from the same region can be dynamically assigned with location aware P2P overlay networks [7]. The super-peer may also delegate actual communication tasks to other peers for better load-balancing.

Here, we have another innovation to further reduce this cost. Instead of receiving all models from other regions, each regional super-peer requests for certain models only. This is accomplished as follows. Every super-peer clusters its data, and sends its cluster information (called Knowledge Spheres, c.f. Section 3.1) to other super-peers. With this knowledge, each super-peer determines the overlap in underlying data space (called Exclusion Spheres, c.f. Section 3.2) between itself

[1] Hence the name Satrap - title for the governor of regional provinces in ancient Persia.

and others, and requests only for models from non-overlapping spaces from an owner super-peer. Upon receiving a request, the owner super-peer gathers support vectors (c.f. Section 3.3) from its model that are relevant to the requester's Exclusion Spheres, and transfers them.

Finally, each super-peer combines all the models received (as before) and then propagates the resultant model to all the peers within its region (c.f. Section 3.4), again with low intra-region cost.

Though this process requires communication of the compact data space representation between regional super-peers, it significantly reduces the propagation of models. In this paper, we omit detailed discussion on failure tolerance and load distribution, and limit our scope to only the data-mining related issues.

3.1 Knowledge Sphere Creation

Unlike test instance propagation where information cannot be compressed or filtered, model propagation in general, allows some form of compression or filtering while enabling representative global models to be constructed.

Since the models of super-peers from different geographical regions may be built on similar data (or data from the same data space), while creating a global model, it is unnecessary for a super-peer to receive *all* information from others. As we do not know a priori what data are overlapping between them, we need a representation of every super-peer's underlying data in the problem space. For this purpose, we propose the use of high dimensional sphere, created from clustering of the data.

After a super-peer cascades the models from its regional peers, we separate the support vectors (SVs) into their separate classes and cluster them. The separation allows more compact clusters to be generated, as SVs from different classes may lie in slightly different input space. The knowledge of these clusters, called the Knowledge Spheres, comprising the centroid (mean of all SVs), radius (maximum distance of any SV in cluster to the centroid), and their density (number of SVs within the cluster definition) is then propagated to all other super-peers.

The reason for using clustering is that it creates groups of neighboring data points which reside close to each other in the problem space, as represented by the high dimensional spheres. Although spheres may not represent the data as well as some other high dimensional shapes such as convex hulls or polygons, they are computationally cheapest to generate and have the best compression ratio (single centroid and radius). We have used agglomerative hierarchical clustering based on single linkage for this task, because it preserves the neighborhood information of the clusters. Another desirable property of this approach is that it produces deterministic results. We also use Euclidean distance as the distance measure for clustering, as it is shown to preserve the neighborhood property between input and feature space [8].

The clusters generated can affect the detection of (non) duplicated data, however we don't know a priori how many clusters would result in the most accurate detection of duplicates. Hence, instead of specifying the number of clusters, peers choose the desired cluster-to-SV ratio R, depending on how many support vectors they have. Note that as the number of clusters reduces, the input space covered by at least one cluster also increases in order to cover the points of the removed clusters. The increase in space covered also includes empty spaces. As the neighborhood area of the input space is correlated to the feature space [8], the feature space covered by the cluster also increases. If we were to filter from such a larger neighborhood (either input or feature space), more points potentially closer to the decision boundary would be filtered, leading to a possibly larger error. It is obvious that as heterogeneity of the regional data increases, the number of clusters required for a compact representation of the data also increases. Moreover, an increase in number of clusters always maintains or improves the cluster compactness (i.e., reduces the intra-cluster distance) but at the cost of addition communication overheard.

3.2 Exclusion Sphere Creation

When a super-peer (say, $r_{requester}$) receives another super-peer's (say, r_{owner}'s) knowledge spheres, it checks if it has as much knowledge about the data space as r_{owner}. It then informs r_{owner} of the knowledge it lacks, so that corresponding knowledge may be transferred. If the number of $r_{requester}$'s SVs falling within the space of an r_{owner} sphere is less than the density of the sphere (times a threshold T), $r_{requester}$ creates a exclusion sphere from those points. The information of the exclusion sphere (centroid, radius, density) along with the corresponding sphere that it overlapped with, is then sent to r_{owner} as part of the data request.

Note that this process is order-dependent. Once $r_{requester}$ has requested information from r_{owner} on a certain data space, it will not request information from another super-peer on an overlapping space, unless of course the latter has significantly larger density. We do not address the order dependency of overlapping checks due to several reasons. Firstly, in order to check the order, a super-peer has to wait for several super-peers to send their knowledge spheres, which is impractical in a dynamic P2P network. Secondly, order dependency only affects performance if there is a quality difference in the data of the different regions, but currently there is no way to verify this difference in quality (unless data points are sent for checking, which is what we want to avoid). Without additional knowledge on the data or communication cost, it would be infeasible to optimize the ordering.

3.3 Gather Relevant SVs

When r_{owner} receives the request, it chooses all SVs that are within the overlapping spheres but outside the exclusion spheres for transfer. It also chooses SVs that lie within the exclusion spheres with a probability of 1 - (number of SVs in exclusion sphere for $r_{requester}$ / number of SVs in exclusion sphere for r_{owner}). We use probabilistic sampling so that SVs within the exclusion sphere are chosen only when the confidence (number of SVs, evidence) of the $r_{requester}$ in the exclusion data space is lower than that of r_{owner}. All the chosen data points are then consolidated and sent to $r_{requester}$. This process marks the end of the cross region data probing and exchange. At this stage, $r_{requester}$ has received models from the entire network if it has requested from all other super-peers. Since the gathering of data is based on the clusters created from the local region cascaded model, it is not order-dependent.

3.4 Global Model Construction and Prediction

Once $r_{requester}$ receives the SVs from r_{owner}, they are merged with the SVs of the local regional cascaded model and the new global cascaded model is built. The new global model can be propagated down-stream to other local regional peers with cheap intra-regional communication. Since every peer now has the global model, all predictions can be made locally without incurring any extra communication cost. In addition, there is no need to wait for predictions from other peers which also saves time. With feedback proposed in [9], the incremental building of the global model at the super-peer is order invariant on the arrival of the exchanged models.

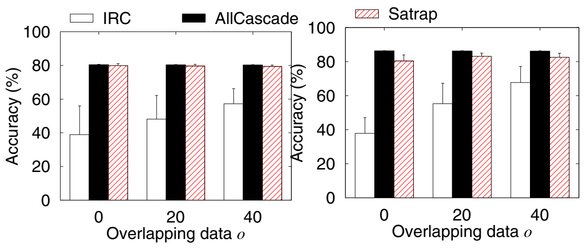
4 Experimental Results

Here, we demonstrate how Satrap exploits data heterogeneity to reduce communication overheads in presence of network heterogeneity, and achieves a good balance between accuracy and communication cost.

We used the multi-class Covertype (581,012 instances, 54 features, 7 classes and 500 peers) and multi-class Waveform (200,000 instances, 21 features, 3 classes and 100 peers) datasets [10]. The datasets were split into ten clusters, each assigned to peers in a separate region to simulate the non-overlapping regional data. To vary data heterogeneity, we overlapped the data in each region with o percent of other regions' data. Experiments were then conducted on these different data distributions. We compared our approach with AllCascade [1], and Individual Regional Cascaded model without cross region data exchange (IRC). All these approaches were implemented in C++ and we used SVM and RSVM implementations from [11,12]. The RBF kernel and penalty cost parameters were selected using the procedure mentioned in [11] and their values are $\gamma = 2, C = 32$ for the Covertype, and $\gamma = 2^{-7}, C = 32$ for the Waveform dataset. For Satrap, the threshold value T is set to 0.75, and the cluster ratio R is set to 0.1. Results were obtained using 10-fold cross validation.

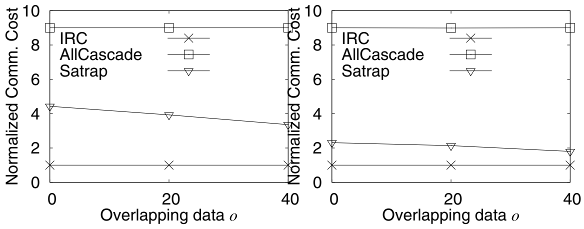
4.1 Performance Evaluation

Figures 2 and 3 present the classification accuracy (in percentage) and communication cost (as a ratio of the total dataset size in the entire network $\times 10^4$) respectively. The plots in Figure 3 are normalized to the cost of IRC which doesn't incur any inter-region costs, and are shown using a conservative 1:1 ratio between intra- and inter-region costs. This ratio can be upto 1:50 in real environments [13], so Satrap's benefits over AllCascade should be amplified.



(a) Multiclass Covertype (b) Multiclass Waveform

Fig. 2. Effect of data overlap on classification accuracy



(a) Multiclass Covertype (b) Multiclass Waveform

Fig. 3. Effect of data overlap on communication cost (normalized to that of IRC)

We varied the percentage o of overlapping data (from other regions) to simulate a varying degree of homogeneity between different regions. From Figure 2, we can see that the varying distribution does not affect the accuracy of AllCascade. However, IRC suffers as the overlap decreases. This is because IRC does not perform any data exchange between different regions, and therefore achieves reasonable accuracy only when data among different regions is homogeneous. Moreover, we observe that the Satrap achieves accuracies close to AllCascade and significantly better than IRC, with only a slight drop as the amount of overlapping data increases. However, this is accompanied by significant savings in communication cost – showing acceptable trade-off between cost and accuracy. We attribute this drop in Satrap's accuracy to the probabilistic sampling for overlapping data space (hence missing out some important data points) which is critical for saving communication cost.

By comparing Figures 2 and 3, we observe that the competing approaches are on the two extremes. IRC has the best accuracy-to-communication cost ratio, but it does not fulfil the criteria to maximize the global accuracy as it does not learn beyond the local region. Observe that the actual accuracy of IRC on average is more than 15% worse than Satrap.

On the other hand, while AllCascade has the highest accuracy, it comes with the lowest accuracy-to-communication cost ratio across all datasets. Satrap closely approximates AllCascade's accuracy while retaining a much superior accuracy-to-communication cost ratio. This ratio significantly improves as the percentage of overlapping data increases. To summarize, we observe that Satrap is able to achieve good accuracy-to-communication cost ratio in most situations.

5 Conclusion

This paper is the first effort that systematically studies the effect of network and data heterogeneity on prediction accuracy and communication cost for learning in P2P networks. Satrap, our network and data heterogeneity aware P2P classification approach, is based on a simple system of information sharing, and lends itself to easy improvement as every module can be fine-tuned depending on knowledge of the domain. Satrap achieves a better accuracy-to-communication cost ratio than existing approaches, and is justified by extensive experiments. The approach also allows users to trade off accuracy for communication cost and vice-versa. In future work, we're looking at how to mitigate the problem of low data overlap, improve the detection of data overlaps and sampling.

References

1. Ang, H.H., Gopalkrishnan, V., Hoi, S.C.H., Ng, W.K.: Cascade RSVM in peer-to-peer networks. In: ECML/PKDD, pp. 55–70 (2008)
2. Bhaduri, K., Wolff, R., Giannella, C., Kargupta, H.: Distributed decision-tree induction in peer-to-peer systems. Statistical Analysis and Data Mining 1(2), 85–103 (2008)
3. Gorodetskiy, V., Karsaev, O., Samoilov, V., Serebryakov, S.: Agent-based service-oriented intelligent P2P networks for distributed classification. In: Hybrid Information Technology, pp. 224–233 (2006)
4. Siersdorfer, S., Sizov, S.: Automatic document organization in a P2P environment. In: Lalmas, M., MacFarlane, A., Rüger, S.M., Tombros, A., Tsikrika, T., Yavlinsky, A. (eds.) ECIR 2006. LNCS, vol. 3936, pp. 265–276. Springer, Heidelberg (2006)
5. Luo, P., Xiong, H., Lü, K., Shi, Z.: Distributed classification in peer-to-peer networks. In: ACM SIGKDD, pp. 968–976 (2007)
6. Breiman, L.: Pasting small votes for classification in large databases and on-line. Machine Learning 36(1-2), 85–103 (1999)
7. Yang, B., Garcia-Molina, H.: Designing a super-peer network. In: ICDE, pp. 49–60 (2003)
8. Shin, H., Cho, S.: Invariance of neighborhood relation under input space to feature space mapping. Pattern Recognition Letters 26(6), 707–718 (2005)
9. Graf, H.P., Cosatto, E., Bottou, L., Dourdanovic, I., Vapnik, V.: Parallel support vector machines: The cascade SVM. In: NIPS, pp. 521–528 (2004)
10. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
11. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
12. Lin, K., Lin, C.: A study on reduced support vector machines. IEEE Transactions on Neural Networks 14(6), 1449–1459 (2003)
13. Touch, J., Heidemann, J., Obraczka, K.: Analysis of HTTP performance. Research Report 98-463, USC/Information Sciences Institute (August 1998)

Player Performance Prediction in Massively Multiplayer Online Role-Playing Games (MMORPGs)

Kyong Jin Shim, Richa Sharan, and Jaideep Srivastava

Department of Computer Science and Engineering
University of Minnesota
200 Union Street Southeast
Minneapolis, MN 55455, USA
{kjshim,srivasta}@cs.umn.edu, sharan.richa@gmail.com

Abstract. In this study, we propose a comprehensive performance management tool for measuring and reporting operational activities of game players. This study uses performance data of game players in EverQuest II, a popular MMORPG developed by Sony Online Entertainment, to build performance prediction models for game players. The prediction models provide a projection of player's future performance based on his past performance, which is expected to be a useful addition to existing player performance monitoring tools. First, we show that variations of PECOTA [2] and MARCEL [3], two most popular baseball home run prediction methods, can be used for game player performance prediction. Second, we evaluate the effects of varying lengths of past performance and show that past performance can be a good predictor of future performance up to a certain degree. Third, we show that game players do not regress towards the mean and that prediction models built on buckets using discretization based on binning and histograms lead to higher prediction coverage.

1 Introduction

Massively Multiplayer Online Role-Playing Games (MMORPGs) are personal computer or console-based digital games where thousands of players can simultaneously sign on to the same online, persistent virtual world to interact and collaborate with each other through their in-game characters. This study is concerned with forecasting of player performance in the game. While many games today provide web and GUI-based reports and dashboards for monitoring player performance, we propose a more comprehensive performance management tool (i.e. player scorecards) for measuring and reporting operational activities of game players. This study uses operational and process-oriented performance data of game players in EverQuest II, a popular MMORPG developed by Sony Online Entertainment, to build performance prediction models for game players. First, we show that variations of PECOTA [2] and MARCEL [3], two most popular baseball home run prediction methods, can be used for game player performance prediction. Second, we evaluate the effects of varying lengths of past performance and show that past performance can be a good predictor of future performance up to a certain degree. Third, we show that game players do not regress towards the mean and that prediction models built on buckets using discretization based on binning and histograms lead to higher prediction accuracy.

Systematic studies of game player performance is expected to yield the following contributions. First, analysis of player performance in different dimensions (i.e. player demographics, archetypes, classes, sub-classes) can help game developers understand whether their games and game characters are being played as intended. Second, benefits for game players are two fold. a) Game players can not only have a view of their past and current performance but also they can have a view of their projected future performance. b) A recommendation engine can be built to recommend character types and tasks to players in order to meet certain objectives (i.e. move up to the next level as fast as possible, play safe by attempting easy tasks, play aggressively by attempting challenging tasks, play tasks that encourage grouping with other players). Third, players can have a view of performances of other players for the purposes of forming quest or raid teams.

2 EverQuest II Game Mechanics

2.1 Point-Scaling System in EverQuest II

In EverQuest II, there is a concept of Ding Points, which is the amount of points one needs to obtain in order to move from one level to the next higher level [4]. For instance, to move from Level 2 to Level 3, one needs to obtain 1,000 points whereas 20,000 points are required to move from Level 73 to 74. The amount of ding points increases as one advances to the next level. As players gain more experience with the game and advance to higher levels, the types of task they can perform increase and the task difficulty also increases. The higher the task difficulty, the higher the potential point gain.

2.2 Tasks in EverQuest II

EverQuest II is rich in types of task players can perform with monster kills being one of the most popular. Monster kills are discussed in details in [1]. In addition to monster kills, other sources of experience points exist in the game such as alternate achievement points (AA) which can be obtained from quests, named mobs, and discovery experience. A player can gain more experience points by having another player mentor him. The mentor levels down to the level of the mentee. The mentee receives a five percent bonus to adventuring experience points.

2.3 Archetypes, Classes, and Sub-classes in EverQuest II

In playing MMORPGs, selection of character type (i.e. archetype, class, subclass, and race) is considered an important decision as it defines the basis of opportunities and choices of roles and tasks within the game [5]. In EverQuest II, there are four archetypes where each archetype consists of three classes each of which in turn consists of two sub-classes [4]. Performance comparisons are discussed in details in [7].

3 Baseball Home Run Prediction

Prediction of future performance of humans has long been studied in various disciplines over the years. Most notably, it has been well studied in sports. Baseball has a long history of record keeping and statistical analyses that dates back to the nineteenth century. Batting average, RBIs, and home runs are some of the many statistics being kept track of today. There exists an enormous amount of public and private interest in the projection of future performance. Major league teams rely on the past statistics of a given player in deciding whether to acquire him or not and for how many seasons under the assumption that his past success is a good indicator of his future success.

PECOTA [2] and MARCEL [3] are widely known methods in baseball home run prediction. PECOTA [2] is considered a very sophisticated method for home run prediction in baseball. For a given ball player at the age of X, the method uses a nearest neighbor analysis of both minor and major league players from the past that exhibited similar performance at age X. It uses the historical performance of these past players to predict the given player's future performance. MARCEL [3] uses data from the three immediate past seasons of a given ball player, and it assigns more weight to more recent seasons. One drawback of this approach is that prediction models solely based on individual players cannot be generalized to the global population. A variation of the MARCEL approach attempts to regress predictions to the global population mean. One drawback of this approach is that prediction models built on the global population can become too coarse.

We consider game player levels in EverQuest II similar to seasons in baseball. Players perform tasks, gain points, and move up to the next level as ball players would attain different types of achievement (i.e. home runs, single, double, triple hits, run batted in, etc.) in each season and proceed to the next season. Unlike in baseball where there is not necessarily a fixed number of home runs, triples, doubles, etc. required to move to the next season, EverQuest II employs a point scaling system where there exists a fixed number of experience points at each level in order to move up to the next level. Because the ding point is a fixed constant, we measure a game player's total play time at each level and uses it as a performance measure in this study.

4 Player Performance Prediction in EverQuest II

In this study, we develop performance prediction models for game players in EverQuest II. The objective is to predict a given player's play time at level i, a future state, based on his past performance at levels $i-1$, $i-2$, and so forth, where performance at any level is measured as the total play time spent at that level. Play time in EverQuest II excludes any idle periods where being idle is defined as any contiguous time blocks of 30 minutes or beyond.

4.1 Methods

MARCEL [3] method uses data from the three immediate past seasons of a given ball player, and it assigns more weight to more recent seasons. One drawback of this approach is that prediction models solely based on individual players cannot be generalized to the global population. A variation of the MARCEL approach attempts to regress predictions to the global population mean. One drawback of this approach is that prediction models built on the global population can become too coarse. Algorithm 1 [7] delineates the steps taken to generate MARCEL-like prediction models for game player performance prediction.

Our preliminary data analysis of the game data reports that play times at each player level exhibit a skewed distribution [7]. EverQuest II game players do not regress towards the mean, and therefore prediction models built under the assumption that they do regress towards the mean will become too coarse and will perform poorly for players whose performances deviate from the mean. To overcome this problem, for a given player, PECOTA [2] uses past performance of those players whose performance patterns are similar to that of the given player.

In this study, we perform data discretization based on two unsupervised techniques, binning and histogram analysis, in order to create buckets of players where all players in a given bucket are termed neighbors. Neighbors share similar performance patterns, and a prediction model is built for each bucket. This is similar to the way PECOTA [2] uses a nearest neighbor analysis to group players into buckets and builds a prediction model for each bucket. Algorithms 2 and 3 [7] delineate the steps taken to create buckets based on binning and histogram analysis, respectively. Algorithms 4 and 5 [7] delineate the steps taken to create MARCEL-like prediction models and Regression-based prediction models, respectively, both using discretization.

4.2 Dataset

The study uses one month worth of performance data from March 1, 2006 to March 31, 2006. The dataset contains over 36 million player-to-task records where over four million of them are monster kills related tasks. The dataset contains 24,571 distinct players across player levels 1 through 70. Since then, Sony Online Entertainment has added an additional ten levels to the game, making 80 the maximum level one can reach.

All of the players and their performance data has been extracted from XP table in the EverQuest II database housed at National Center for Supercomputing Applications (NCSA) at the University of Illinois. The dataset contains at the minimum the following information about game players: character id, character sub-class, race, task, timestamp of task completion, group size (whether a given character grouped with one or more other characters), average group level (if a given character played with one or more other characters, this value represents the average of player levels of all players involved in that group), experience points, location (location in which the task was completed).

4.3 Evaluation

In prediction (i.e. regression, time series analysis, etc.), a common practice has been to specify coverage probabilities by convention, 90%, 95%, and 99% being typical choices. A previous study [6] reports that academic writers concentrate on 95% intervals while practical forecasters prefer 50% intervals. In this study, we compute prediction coverage at varying confidence intervals at 80% and 90%. Algorithm 6 [7] delineates the steps taken to compute prediction coverage.

5 Experiments and Results

5.1 Past Performance as Indicator of Future Performance

A series of experiments have consistently shown that the three immediate past levels contribute the most to the prediction of a player's future performance. Extending beyond the three immediate past levels does not positively contribute to prediction coverage. One possible explanation might be that game players, in playing tasks such as monster kills in EverQuest II, do not tend to degrade in their performance suddenly, and therefore, a given player's performance at the most recent level $(i-1)$ should be most informative about his performance at the current level (i). However, this may not necessarily be true in all cases such as when a player all of a sudden decides to attempt monsters whose levels are far beyond average, in which case, the player's performance at the current level may degrade due to the fact that his skill level is suddenly not matching the task difficulty. Additionally, we try a variety of weighting schemes for use with MARCEL [3] approach. Broadly, weighting functions are categorized into 1) even weight distribution and 2) decaying weight distribution. The former assigns an equal amount of confidence to each of the past levels whereas the latter assigns more weight to more immediate past levels. Our findings suggest that with the three immediate past levels, both even weight distribution and decaying weight distribution produce comparative results.

5.2 Discretization Improves Prediction Coverage

Given the dataset used in our analysis, our findings suggest that the bucket number of six leads to high prediction coverage. In some player levels though we observe that a bucket number slightly lower or higher than six leads to even higher prediction coverage.

Our results show that discretization using binning and histogram analysis leads to higher prediction coverage overall across all 70 player levels where the number of buckets is six. Figure 1 shows that MARCEL [3] approach produces

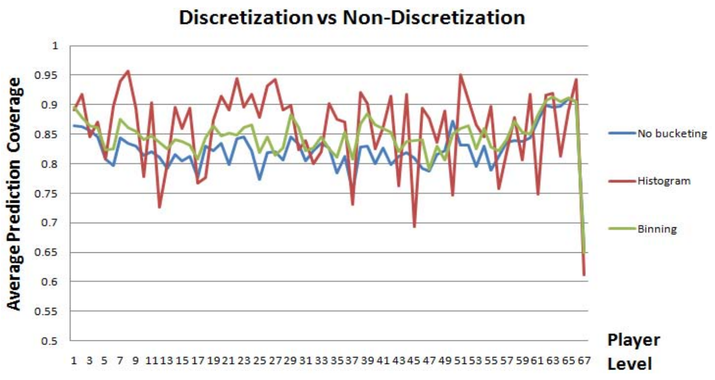


Fig. 1. Discretization Improves Prediction Coverage (MARCEL approach)

an average prediction coverage of 82.4% whereas the same approach employing binning produces 84.7% and that employing histogram analysis produces 86% prediction coverage (confidence interval of 80%). Figure 2 shows results consistent with MARCEL approach where the base linear regression model produces an average prediction coverage of 83.2% whereas the model employing binning produces 85% and that employing histogram analysis produces 85.7% prediction coverage (confidence interval of 80%).

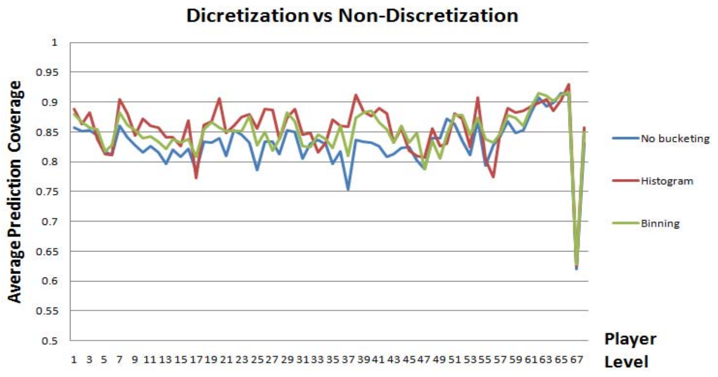


Fig. 2. Discretization Improves Prediction Coverage (Linear Regression)

5.3 Comparison of Prediction Models

Figure 3 shows prediction coverage computed at confidence interval of 80%. MARCEL [3] approach in combination with histogram-based discretization performs the best while all other schemes produce results that are comparative to that of MARCEL [3] approach.

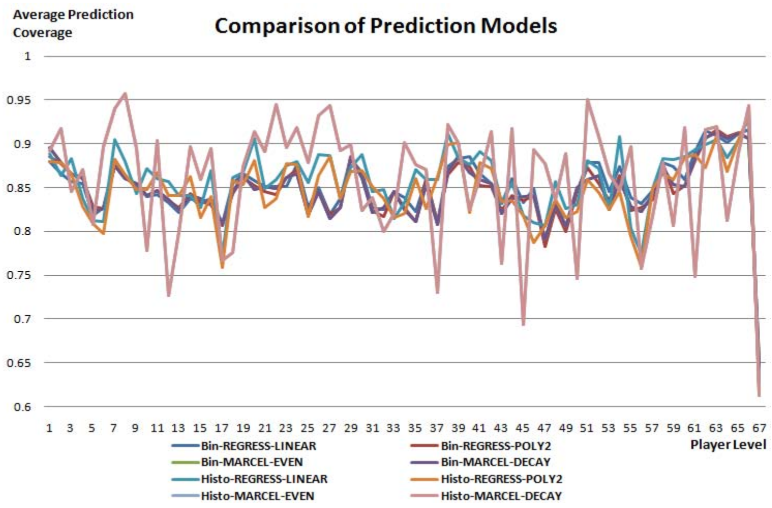


Fig. 3. Comparison of Prediction Models (80% Interval)

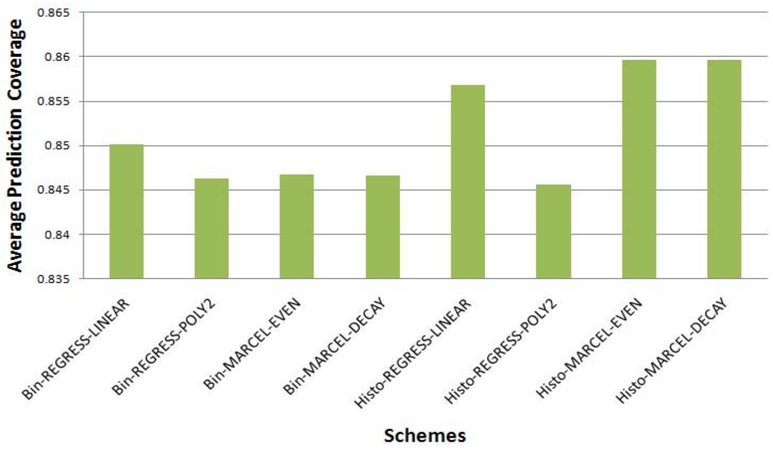


Fig. 4. Comparison of Prediction Models (80% Interval)

Figure 4 charts the average prediction coverage computed at confidence interval of 80% across 70 player levels. MARCEL [3] approach in combination with histogram-based discretization performs the best while all other schemes produce results that are comparative to that of MARCEL [3] approach.

Figure 5 shows prediction coverage computed at confidence interval of 90%. Linear regression model in combination with binning-based discretization performs the best while all other schemes produce results that are comparative to that of linear regression model.

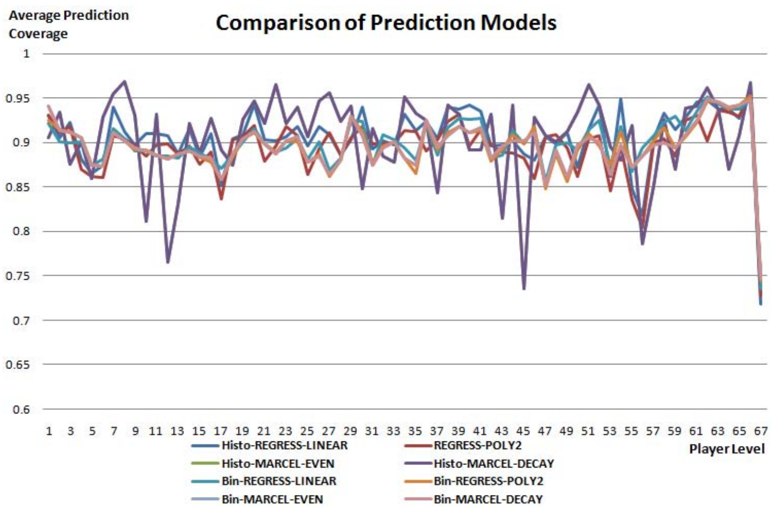


Fig. 5. Comparison of Prediction Models (90% Interval)

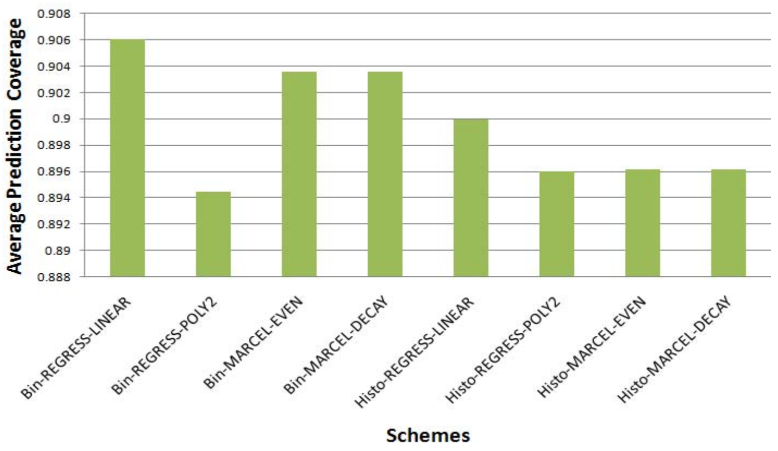


Fig. 6. Comparison of Prediction Models (90% Interval)

Figure 6 charts the average prediction coverage computed at confidence interval of 90% across 70 player levels. Linear regression model in combination with binning-based discretization performs the best while all other schemes produce results that are comparative to that of linear regression model.

Our prediction models capture information essential about the relationship between progression of player level and progression of player performance (as a function of play time) over a range of three player levels. Our results consistently show that the relationship is linear to a certain extent. This trend is observed across all 70 player levels.

6 Conclusion

In this paper, we show that variations of PECOTA [2] and MARCEL [3], two most popular baseball home run prediction methods, can be used for game player performance prediction. MARCEL approach in combination with bucketing inspired from PECOTA approach leads to high prediction coverage. The method uses data from the three immediate past levels and assigns more weight to more recent levels. In game player performance prediction, our findings suggest that the results from even weight distribution and decay weight distribution are comparative. To account for an observation that game players in EverQuest II do not regress towards the mean in terms of their play times, prediction models are built on buckets using discretization based on binning and histograms. This approach leads to higher prediction coverage. Further, we build regression-based models and show that the relationship between progression of player level and progression of player performance (as a function of play time) over a range of time is linear to a certain extent. The regression-based models produce prediction coverage comparative to that of existing methods.

Prediction models we propose in this study are expected to be a useful addition to many existing player performance monitoring tools by providing a projection of a given player's future performance given his past performance. Game player performance data such as that of EverQuest II is rich of not only outcome data (i.e. number of monsters killed, number of experience points gained, number of deaths occurred, number of quests completed in a given time duration) but also process data, from which we can construct a progression of a given player's performance at any given time point. Existing player performance monitoring tools can be greatly enhanced to dynamically capture player performance progression, provide instant feedback on player's progress, and recommend tasks tailored towards a given player's objectives of playing the game (performance-oriented tasks vs. social activity-oriented).

7 Future Directions

An extension to the current work involves investigating model dynamics by examining the balancing of past consistency with advancing player level. An issue arises when a player performs way below the average for a couple of levels and springs back up to a very good performance. All of the prediction models discussed in this study so far lack the ability to integrate such dynamics into prediction. Another extension to the present study seeks to define performance in many dimensions of different granularity levels (i.e. task types, archetypes, classes, sub-classes, races, roles, etc.). For instance, the present study defines performance as a function of play time or active time. Another measure of performance is the level of consistency and commitment. Results from such analyses can reveal player behavioral patterns indicative of player churning. Yet another addition to this study is to leverage a variety of social networks in EverQuest II (i.e. housing network, trust network, raid group network, and guild network) to measure the impact of social interactions on player performance.

Acknowledgments

The research reported herein was supported by the National Science Foundation via award number IIS-0729421, and the Army Research Institute via award number W91WAW-08-C-0106. The data used for this research was provided by the SONY corporation. We gratefully acknowledge all our sponsors. The findings presented do not in any way represent, either directly or through implication, the policies of these organizations. Acknowledgements are also due to the members of the University of Minnesota, Northwestern University, University of Southern California, University of Illinois, University of Michigan, and University of Chicago.

References

1. Shim, K.J., Ahmad, M., Pathak, N., Srivastava, J.: Inferring Player Rating from Performance Data in Massively Multiplayer Online Role-Playing Games (MMORPGs), cse. In: International Conference on Computational Science and Engineering, vol. 4, pp. 1199–1204 (2009)
2. Silver, N.: Introducing PECOTA, pp. 507–514. Baseball Prospectus, Hyattsville (2003)
3. Tango, T.: Marcel The Monkey Forecasting System. Tangotiger .net, http://www.tangotiger.net/archives/stud0346.shtml (March 10, 2004)
4. Prima Development. Everquest II: Official Strategy Guide (Prima Official Game Guides)
5. http://pc.gamespy.com/pc/everquest-ii/564411p1.html
6. Granger, C.W.J.: Can We Improve the Perceived Quality of Economic Forecasts? Journal of Applied Econometrics 11, 455–473 (1996)
7. Shim, K.J., Sharan, R., Srivastava, J.: Player Performance Prediction in Massively Multiplayer Online Role-Playing Games (MMORPGs), Technical Report 10-003, Department of Computer Science and Engineering, University of Minnesota

Relevant Gene Selection Using Normalized Cut Clustering with Maximal Compression Similarity Measure

Rajni Bala[1], R.K. Agrawal[2], and Manju Sardana[2]

[1] Deen Dayal Upadhyaya College, University of Delhi, Delhi, India
[2] School of Computer and System Science, Jawaharlal Nehru University, New Delhi, India

Abstract. Microarray cancer classification has drawn attention of research community for better clinical diagnosis in last few years. Microarray datasets are characterized by high dimension and small sample size. To avoid curse of dimensionality good feature selection methods are needed. Here, we propose a two stage algorithm for finding a small subset of relevant genes responsible for classification in high dimensional microarray datasets. In first stage of algorithm, the entire feature space is divided into k clusters using normalized cut. Similarity measure used for clustering is maximal information compression index. The informative gene is selected from each cluster using t-statistics and a pool of non redundant genes is created. In second stage a wrapper based forward feature selection method is used to obtain a set of optimal genes for a given classifier. The proposed algorithm is tested on three well known datasets from Kent Ridge Biomedical Data Repository. Comparison with other state of art methods shows that our proposed algorithm is able to achieve better classification accuracy with less number of features.

Keywords: Cancer Classification, Microarray, Normalized Cut, Representative Entropy, Gene Selection.

1 Introduction

DNA microarrays have provided the opportunity to measure the expression levels of thousands of genes simultaneously. One of the most common application of microarray is to classify the samples such as healthy versus diseased by comparing the gene expression levels. Microarray data which is characterized by high dimension and small sample size suffers from curse of dimensionality[1]. For better classification there is a need to reduce the dimension. In general, among thousands of genes(features) which are monitored simultaneously only a fraction of them are biologically relevant. Therefore, efficient feature selection methods are needed to identify a set of discriminatory genes that can be used for effective class prediction and better clinical diagnose. In literature, various feature selection methods have been proposed. These methods broadly fall into two categories[2]: filter and wrapper methods. Most filter methods independently measure the importance of features without involving any classifier. So, they may not select the most relevant set of features for the learning algorithm. Also, the features set selected by filter methods may contain correlated(redundant) features which may degrade the performance of classifier. On the other hand, wrapper methods directly use the classification accuracy of some classifier as the evaluation criteria. They tend to find features better suited to the predetermined learning algorithm resulting in better performance. But, they are computationally more expensive . The conventional wrapper methods are hard to apply directly to high dimensional datasets as they require large computation time. Reducing the search space for wrapper methods will decrease the computation time. This can be achieved by first selecting a reduced set of non-redundant features from the original set of features without losing any informative feature.

In this paper, a novel two-stage approach is proposed to determine a subset of relevant and non-redundant genes for better cancer classification. Our approach first groups correlated genes and then select one informative gene from each one of these groups to reduce redundancy. This requires partitioning of the original gene set into some distinct clusters so that the genes within a cluster are highly similar(correlated) while those in different clusters are dissimilar. At the second stage a Sequential Forward Feature Selection(SFFS) method is applied to select a smaller set of discriminatory genes which can provide maximum classification accuracy.

This paper is organized as follows. Section 2 describes related work. In section 3 we present our proposed algorithm for selecting a set of informative and non-redundant genes. Experimental results on some well-known datasets are presented in Section 4. Section 5 contains conclusions.

2 Related Work

In order to achieve better classification of high dimensional microarray data, we need to determine a smaller set of discriminatory genes from a given set of genes without loosing any information. In literature, many gene selection methods have been proposed which are based on a gene ranking that assigns a score for each gene which approximates the relative strength of the gene. These methods return a set of top ranked genes and classifier is built on these genes. Among them, Golub et. al.[3] selected top genes using measure of correlation which emphasizes that a discriminatory gene must have close expression levels in samples within a class, but significantly different expression levels in samples across different classes. Other approaches that adopt the same principle with modifications and enhancements include[4] and [5]. Using ranking method, one cannot select a smallest set of discriminatory genes as the selected subset may contain many correlated genes. Few wrapper based approaches are also suggested in literature which works better for small and middle dimensional data. However they cannot be applied directly on high dimensional microarray dataset as it is computationally expensive. We can overcome this by determining a smaller set of genes for wrapper approach. This is possible if we can group correlated or similar genes into clusters and then select a gene from each cluster which can provide us a reduced set of independent and informative genes.

In literature clustering has been employed for grouping correlated or similar genes. Many diverse clustering techniques have been suggested in literature. The most widely used techniques include hierarchical[6], k-means clustering[7] and Self-organized- maps(SOM)[8]. Each one of them is associated with advantages and disadvantages. Shi and Malik[9] have proposed an efficient normalized cut(NCUT) method based on graph theoretic approach for image segmentation. The normalized cut criterion measures both the total dissimilarity between the different groups as well as total similarity with in the groups. This can also be used for clustering of correlated genes in microarray data. In NCUT a given graph G=(V, E), where $v_i \epsilon$ V represents a gene and $e(v_i, v_j) \epsilon$ E represents similarity between two genes v_i and v_j, is divided into two disjoint sets A and B. For partitioning of the genes into A and B, the capacity of the normalized cut, Ncut is defined as

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \tag{1}$$

where $cut(A,B) = \Sigma_{u \epsilon A, v \epsilon B} w(u,v)$ and $assoc(A,V) = \Sigma_{u \epsilon A, t \epsilon V} w(u,t)$

To determine a better partition of a cluster the value of Ncut should be minimized which is a NP-hard problem. Shi and Malik[9] have shown that this problem can be reformulated as eigenvalue problem which is given by

$$D^{-1/2}(D-W)D^{-1/2}x = \lambda x \tag{2}$$

It has been shown by Shi and Malik[9] that second smallest eigenvector of the above generalized eigenvalue system is the real valued solution to our minimum normalized cut problem. Hence, the second smallest eigenvector can be used to partition the original cluster into two clusters.

In general euclidean distance and Pearsons correlation are used as the distance or similarity measure for clustering. However, euclidean distance is not suitable to capture functional similarity such as positive and negative correlation, and interdependency[10]. It is also pointed out that it is suitable only for a data which follows a particular distribution[11]. On other hand, Pearson coefficient is not robust to outliers and it may assign a high similarity score to a pair of dissimilar genes[12]. Also both these measures are sensitive to scaling and rotation. A similarity measures called maximal information compression index[13] is suggested in literature for measuring redundancy between two features. Given two random variables x_1 and x_2, the maximal information compression index $\lambda_2(x_1, x_2)$ is defined as

$$\lambda_2(x_1,x_2) = \frac{\sigma_1 + \sigma_2 + \sqrt{((\sigma_1+\sigma_2)^2 - 4\sigma_1\sigma_2(1-\rho(x_1,x_2)^2)}}{2} \tag{3}$$

where σ_1, σ_2 are the variance of x_1, and x_2 respectively and $\rho(x_1, x_2)$ is the correlation between x_1 and x_2.

The value of λ_2 is zero when the features are linearly dependent and increases as the amount of dependency decreases. The measure λ_2 possesses several desirable properties such as symmetry, sensitivity to scaling and invariance to rotation which are not present in the commonly used euclidean distance and correlation coefficient.

Further splitting of a cluster, from a set of available clusters, can be decided on the basis of representative entropy measure. Representative entropy measures the amount of redundancy among genes in a given cluster. For a cluster containing p genes with covariance matrix Σ, representative entropy, H_R of a cluster is given by

$$H_R = -\Sigma_{l=1}^{p} \overline{\lambda_l} log(\overline{\lambda_l}) \tag{4}$$

where $\overline{\lambda_l} = \frac{\lambda_l}{\Sigma_{l=1}^{p}\lambda_l}$ and $\lambda_l, l = 1, 2, \ldots, p$ are the eigen values of the matrix Σ.

H_R attains a minimum value(zero) when all the eigenvalues except one are zero, or in other words when all the information is present along a single direction. If all the eigenvalues are equal, i.e. information is equally distributed among all the genes, H_R is maximum. High value of H_R represents low redundancy in the cluster. Since we are interested in partitioning the original subspace into homogeneous clusters, each cluster should have low H_R. So we split a cluster which has maximum H_R among a given set of clusters as it contains more non-redundant genes.

3 Proposed Method for Gene Selection

Here we propose a two stage algorithm to select a set of discriminatory genes to achieve better classification. Our proposed algorithm consists of two phases. The first phase involves partitioning of the original gene set into some distinct clusters so that the genes within a cluster are highly correlated to each other while those in different clusters are less correlated. The similarity measure used in NCUT clustering algorithm is maximal information compression index. We have used a hierarchical clustering in which we start with a single cluster. We split the original cluster into two clusters such that the normalized cut value is minimized. To determine which candidate cluster to further partition from the existing set of clusters, we have used representative entropy. The cluster with the maximum H_R(low redundancy) is partitioned. This process is repeated till we get the required number of clusters. Representative gene from each cluster is chosen using t-statistics. In the second phase a Sequential Forward Feature selection(SFFS) method is applied to select a smaller set of discriminatory genes which provides maximum accuracy. The criterion used in the SFFS is the accuracy of the classifier. The outline of the proposed algorithm is the following:

```
Proposed Algorithm
Input : Initial Set of genes, Class Labels C, Classifier M,
        Cluster_Size
PHASE  1 // to determine a subset of relevant and independent
            genes S
```

```
1. Intialization : Set G=initial set of genes ;
2. S = empty set; No_of_clusters=2; /*Set of Selected Attributes*/
3. Calculate the Similarity Matrix W using Maximal information compression index.
4. Define D where $D(i) = \sigma_j w(i, j)$
5. Solve eigenvalue problem $D^{-1/2}(D - W)D^{-1/2}x = \lambda$
6. Use the eigenvector with second smallest eigenvalues to divide the original cluster C into two clusters.
7. While (no_of_clusters$\leq$Cluster_Size)
8. Begin
9. For each cluster calculate the representative entropy $H_R$
10. Choose the Cluster $C_i$ having the maximum entropy
11. Repeat step (3)-(6) for Cluster $C_i$
12. No_of_clusters=No_of_clusters+1
13. End
14. For each cluster
15. Find the informative gene $g_i$ from cluster $C_i$ using t-statistics
16. S=S U $g_i$
```

```
PHASE 2 // to determine subset of genes which provides max accuracy
```

```
1.Initialization R=empty set
2.For each $x_j \in S$ calculate classification accuracy for classifier M.
3.$[x_k, max\_acc] = max_j \ Classification\_accuracy(x_j)$;
4.$R = R \cup x_k; S = S - x_k; R\_min = R$
5. For each $x_j$ calculate classification_accuracy of $S \cup x_j$ for classifier M
6. $[x_k, max\_acc] = max_j \ Classification\_accuracy(x_j)$;
7. $R = R \cup x_k; S = S - x_k$
8. If new_max_acc $\geq$ max_acc then R_min=R;max_acc=new_max_acc;
9. Repeat 5-9 until max_acc=100 or S = empty set
10. Retum R_min, max_acc
```

4 Experimental Setup and Results

To test the effectiveness of our proposed algorithm, we have carried out experiments on three well known datasets from Kent Ridge Biomedical Data Repository[14]. The details of these datasets are given in Table 1. Datasets are normalized using Z-score before carrying out experiments.

Table 1. Datasets Used

Dataset	Samples	Genes	Classes
Colon	62	2000	2
SRBCT	83	2308	4
Prostate	102	5967	2

Table 2. Maximum classification accuracy along with number of genes for different classifiers using different cluster size methods

No.of Clusters	LDC	QDC	KNN	SVM
30	93.54(18)	91.93(24)	95.16(13)	95.16(14)
40	91.93(4)	93.54(8)	95.16(6)	93.54(5)
50	91.93(4)	95.16(6)	96.77(11)	95.16(10)
60	98.38(32)	95.16(7)	95.16(8)	96.77(19)

a. Colon dataset

No.of Clusters	LDC	QDC	KNN	SVM
30	97.59 (20)	96.38 (10)	100 (7)	100 (4)
40	100 (31)	97.59 (11)	100 (6)	100 (4)
50	100 (33)	97.59 (11)	100 (6)	100 (5)
60	98.79 (9)	97.59 (12)	100 (6)	100 (6)

b. SRBCT dataset

No.of Clusters	LDC	QDC	KNN	SVM
30	93.13 (3)	96.07 (3)	98.03 (7)	97.06 (14)
40	96.07 (8)	96.07 (3)	96.07 (3)	99.01 (15)
50	96.07 (8)	96.07 (3)	96.07 (3)	98.03 (17)
60	97.05 (5)	97.05 (19)	99.01 (7)	96.07 (3)

c. Prostate dataset

Table 3. Comparison of Maximum Classification accuracy and number of genes selected with other state of art methods

SRBCT	PROSTATE	COLON
Proposed Method 100(4)	Proposed Method 99.01(7)	Proposed method 98.38(32)
GS2+SVM[4] 100(96)	GAKNN[17] 96.3(79)	PSO+ANN[4] 88.7
GS1+SVM[4] 98.8(34)	BIRS[18] 91.2(3)	Yuechui and Yao[20] 90.3
Chos+SVM[4] 98.8(80)		BIRSW[18] 85.48(3.50)
Ftest + SVM[4] 100(78)		BIRSF[18] 85.48(7.40)
Fu and Liu[15] 100(19)		
Tibsrani[19] 100(43)		
Khan[16] 100(96)		

Genes are clustered using NCUT based on maximal information compression index as similarity measure. From each cluster the most informative gene is selected using t-statistics. After collecting a pool of genes, a Forward Feature Selection method is applied to get a sub-optimal set of genes which provides maximum classification accuracy. Classification accuracy is calculated using leave-one-out cross validation. The different classifiers used in our experiments are linear discriminant classifier(LDC), quadratic discriminant classifier(QDC), k-nearest neighbor(KNN) and support vector machine(SVM). For KNN the optimal value of k is chosen. Linear kernel is used in SVM. The experiment was conducted for different cluster sizes. The cluster sizes considered in our experiments are 30, 40, 50 and 60. Table 2 depicts the maximum classification accuracy along with the number of genes obtained by our proposed algorithm for different cluster sizes. We can observe the following from Table 2:

1. For Colon dataset a maximum accuracy of 98.38% is achieved with 32 genes for LDC classifier. The maximum accuracy of 96.77% is achieved for KNN and SVM with 11 and 19 genes respectively. For QDC a maximum accuracy of 95.16% is achieved with 6 genes.
2. For SRBCT dataset maximum classification accuracy of 100% is achieved for LDC, KNN and SVM with 31 , 6 and 4 genes respectively. For QDC a maximum accuracy of 97.59% is achieved with 11 genes.
3. For prostate dataset maximum classification accuracy of 99.01% is achieved for KNN and SVM with 7 and 15 genes respectively. For QDC and LDC a maximum accuary of 97.05% is achieved with 19 and 5 genes respectively.
4. The performance of KNN is better in terms of number of genes in comparison to other classifiers LDC, QDC and SVM for all three data sets.

It is observed that our proposed algorithm is able to achieve a high classification accuracy with small number of genes. In Table 3, we have also compared performance of our proposed method in terms of classification and number of genes with some already existing gene selection methods in literature[15],[16],[17],[18], [19],[4]and [20]. From Table 3, it can be observed that the performance of our proposed algorithm is significantly better in terms of both classification accuracy and number of genes selected.

5 Conclusion

In this paper, we have proposed a two stage algorithm for finding a small subset of discriminatory genes responsible for classification in high dimensional microarray datasets. The first stage involves partitioning of the original gene set into some distinct subsets or clusters so that the genes within a cluster are highly correlated to each other while those in different clusters are less correlated. We have used NCUT clustering algorithm which is based on graph theoretic approach and requires computation of similarity measures between genes. We have used a novel similarity measure maximal information compression index which is not used for microarray datasets earlier. Most informative gene from each cluster is then selected to create a pool of non-redundant genes. The size of this set is significantly small which allows us to use a wrapper approach at the second stage. The use of wrapper method at the second stage gives a smaller subset of genes which provides better classification accuracy. Experimental results show that our proposed method is able to achieve a better accuracy with a small number of genes. Comparisons with other state of art methods show that our proposed algorithm is able to achieve better or comparable accuracy with less number of genes with all the three datasets.

References

1. Bellman, R.: Adaptive Control Processes. A Guided Tour. Princeton University Press, Princeton (1961)
2. Guyon, I., Elisseeff, A.: An Introduction to Variable and feature Selection. Journal of Machine Learning Research (3), 1157–1182 (2003)

3. Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P.: Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. Science 286, 531–537 (1999)
4. Yang, K., Cai, Z., Li, J., Lin, G.H.: A stable gene selection in microarray data analysis. BMC Bioinformatics 7, 228 (2006)
5. Cho, J., Lee, D., Park, J.H., Lee, I.B.: New gene selection for classification of cancer subtype considering within-class variation. FEBS Letters 551, 3–7 (2003)
6. Eisen, M.B., Spellman, T.P.: Cluster analysis and display of genome-wide expression patterns. Proc. Natl. Acad. Sci. USA 95(25), 14863–14868 (1998)
7. Tavazoie, S., Huges, D., Campbell, M.J., Cho, R.J., Church, G.M.: Systematic determination of genetic network architecture. Nature Genet., 281–285 (1999)
8. Kohonen, T.: Self-organizing maps. Springer, Berlin (1995)
9. Shi, J., Malik, J.: Normalized Cuts and Image Segmentation. IEEE Transactions on Pattern analysis and machine Intelligence 22(8), 888–903 (2000)
10. Jiang, D., tang, C., Zhang, A.: Cluster Analysis for gene expression data: A survey. IEEE Trans. Knowledge and Data Eng. 16, 1370–1386 (2004)
11. Yu, J., Amores, J., Sebe, N., Tian, Q.: Toward Robust Distance Metric analysis for Similarity Estimation. In: Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition (2006)
12. Heyer, L.J., Kruglyak, S., Yooseph, S.: Exploring Expression Data: identification and analysis of coexpressed genes. Genome Research 9, 1106–1115 (1999)
13. Mitra, P., Murthy, C.A., Pal, S.: Unsupervised feature selection using feature similarity. IEEE Trans. Pattern Analysis and Machine Intelligence 24(3), 301–312 (2002)
14. Kent Ridge Biomedical Data Repository, `http://datam.i2r.a-star.edu.sg/datasets/krbd/`
15. Fu, L.M., Liu, C.S.F.: Evaluation of gene importance in microarray data based upon probability of selection. BMC Bioinformatics 6(67) (2005)
16. Khan, J., Wei, S., Ringner, M., Saal, L.H., Ladanyi, M., Westermann, F.: Classification and diagnosis prediction of cancers using gene expression profiling and artificial neural networks. Nat. Med. 7, 673–679 (2001)
17. Li, L., Weinberg, C.R., Darden, T.A., Pedersen, L.G.: Gene Selection for sample classification based on gene expression data: Study of sensitivity to choice of parameters of the GA/KNN method. Bioinformatics 17(12), 1131–1142 (2001)
18. Ruiz, R., Riqueline, J.C., Aguilar-Ruiz, J.S.: Incremental wrapper based gene selection from microarray data for cancer classification. Pattern Recognition 39(12), 2383–2392 (2006)
19. Tibsrani, R., Hastie, T., Narasimhan, B., Chu, G.: Diagnosis of multiple cancer types by shrunken centriods of gene expression. Proc. Natl Acad. Sci., USA (99), 6567–6572 (2002)
20. Yuechui, C., Yaou, Z.: A novel ensemble of classifiers for microarray data classification. Applied Soft computing (8), 1664–1669 (2008)

A Novel Prototype Reduction Method for the K-Nearest Neighbor Algorithm with $K \geq 1$

Tao Yang, Longbing Cao, and Chengqi Zhang

Faculty of Engineering and Information Technology
University of Technology, Sydney, Australia

Abstract. In this paper, a novel prototype reduction algorithm is proposed, which aims at reducing the storage requirement and enhancing the online speed while retaining the same level of accuracy for a K-nearest neighbor (KNN) classifier. To achieve this goal, our proposed algorithm learns the weighted similarity function for a KNN classifier by maximizing the leave-one-out cross-validation accuracy. Unlike the classical methods PW, LPD and WDNN which can only work with $K = 1$, our developed algorithm can work with $K \geq 1$. This flexibility allows our learning algorithm to have superior classification accuracy and noise robustness. The proposed approach is assessed through experiments with twenty real world benchmark data sets. In all these experiments, the proposed approach shows it can dramatically reduce the storage requirement and online time for KNN while having equal or better accuracy than KNN, and it also shows comparable results to several prototype reduction methods proposed in literature.

1 Introduction

We consider a general classification problem with $C(\geq 2)$ classes and n training instances. Each training instance consists of measurements $\mathbf{x} = (x_1, \ldots, x_d)^T$ on d features and a known class label $y = \{1, 2, \ldots, C\}$. The training data set can be represented in the form of $\Omega = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$. The goal of a classification task is to correctly predict the class label of a query $\mathbf{q} \in \Re^d$.

The K-nearest neighbor (KNN) method is a simple and appealing approach to this problem [3]. The KNN algorithm is also known as nearest neighbor (NN) algorithm for $K = 1$. It is well known that the asymptotic error rate of the NN rule is never more than twice the Bayes rate [1]. However, the major outstanding drawback of the KNN algorithm is that the whole training data set must be stored in memory to be used in the test phase. To identify the K nearest neighbors in the test phase, the distances between $\mathbf{q}$ and all training instances $\mathbf{x}_i$ $(i = 1, \ldots, n)$ must be computed. This can result in a prohibitively large storage requirement and slow testing speed. (Note that the term 'testing speed' and 'online speed' can be used interchangeably.) Also, the presence of noisy instances (i.e., those with errors in the feature vector or class label, or those not representative of typical cases) can also degrade the generalization performance of KNN.

Prototype reduction techniques are concerned with reducing the number of training vectors (prototypes) to be used, which can reduce the storage requirement and increase the testing speed simultaneously. Some prototype reduction methods identify the optimal subset of the representative instances from the original data, while the other approaches generate an entirely new set of objects as the artificial prototypes. Suppose the size of the stored training set is n, then the testing time for KNN to classify one query is $O(dn^2)$. Hence, the reduction in storage requirement can subsequently enhance the testing speed of a KNN classifier. A comprehensive survey of the prototype reduction methods for KNN can be found in [12].

A recent and very promising approach for prototype reduction, called Weighted Distance Nearest Neighbor (WDNN) [2], is based on retaining the informative instances and learning their weights for classification. The WDNN algorithm assigns a weight $w_i(\geq 0)$ to each training instance $\mathbf{x}_i$ at the training phase. Only the training instances and the corresponding weights with $w_i > 0$ will be retained (as the prototypes) in the test phase. Although only a fraction of the training set is retained, the generalization performance of WDNN can be equal to or even better than NN. To achieve this goal, the weights w_i are determined by maximizing the leave-one-out cross-validation (LOOCV) accuracy. At each iteration of a LOOCV procedure, each training instance $\mathbf{x}_i$ is regarded as the query and its class label is predicted by using all the other training instances. This procedure is repeated for $i = 1, \ldots, n$. The WDNN algorithm is a hill climbing optimization technique where each w_i is updated by assuming all the other weights w_j $(j \neq i)$ are given and fixed, and the optimal w_i is determined by considering the threshold value for which $\mathbf{x}_i$ will be the nearest neighbor of the other training instances. In [2], it has been shown that the WDNN algorithm can reduce, on average, more than 80% of the training set while retaining or improving the generalization accuracy of a NN classifier over several real data sets. In the same paper, the WDNN algorithm has also been shown to outperform several benchmarking prototype methods including A-NN [11], PW [7] and LPD [6].

Although the WDNN algorithm is well formulated and shows encouraging performance in practice, it can only work with $K = 1$. (Similarly, PW [7] and LPD [6] also work with $K = 1$ only). However, it has been shown that the best value of K is neither too large nor too small, and setting $K > 1$ can reduce the sensitivity to noise and smooth the decision boundaries (see [1] and [3]). In the case of $K = 1$, being a nearest neighbor of an instance and classifying this instance to its class are the same thing. For $K > 1$, these two things are different because the decision on the classification of a query is determined by K nearest neighbors. This fact along with the possible tied votes make the weight learning for $K > 1$ complicated and difficult.

In this paper, we extend the WDNN algorithm to a general weight learning algorithm for KNN with $K \geq 1$. Naturally, the developed algorithm is dubbed the Weighted Distance K Nearest Neighbor (WDKNN) algorithm. In fact, WDNN is a special case of WDKNN as NN is a special case of KNN. As with WDNN, the WDKNN algorithm iteratively updates the instance weights by maximizing the LOOCV accuracy. However, the crucial difference between WDKNN and WDNN is that the weights returned by WDKNN are derived from an explicit objective function and model of the decision function. In addition, it has been shown that the optimal weights can be determined by using a subset of the training set only and the difficulties caused by $K > 1$ have been successfully resolved by considering two threshold values in WDKNN.

The rest of this paper is organized as follows. In Section 2, the KNN classification rule combined with instance weights is introduced. Section 3 presents our proposed WDKNN algorithm. The experiment results are given in Section 4. Some concluding remarks are given in Section 5.

2 KNN Classification with Weighted Instances

In this section, we introduce the KNN classification rule with the specified instance weights w_i $(i = 1, \ldots, n)$. Here, we assume that these weights have already been learned by the WDKNN algorithm and we will present how they are learned in the following section.

The K nearest neighbors of $\mathbf{q}$ are found by using the weighted similarity between $\mathbf{q}$ and $\mathbf{x}_i$:

$$\mu_w(\mathbf{q}, \mathbf{x}_i) = w_i \cdot \mu(\mathbf{q}, \mathbf{x}_i), \qquad i = 1, \ldots, n, \tag{1}$$

where

$$\mu(\mathbf{q}, \mathbf{x}_i) = 1 - D(\mathbf{q}, \mathbf{x}_i)/D_{\max}, \tag{2}$$

$$D(\mathbf{q}, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{d} (q_j - x_{ij})^2}. \tag{3}$$

Here, $D_{\max}$ is the maximum possible distance between two training instances in the feature space, which is used to allow $\mu(\mathbf{q}, \mathbf{x}_i)$ to fall into the interval of $[0, 1]$. That is, $D_{\max} = \sqrt{\sum_{j=1}^{d} \Delta_j^2}$ and Δ_j represents the difference between maximum and minimum values of feature j. If we denote the neighborhood around $\mathbf{q}$ by $N(\mathbf{q})$ and the Kth largest value of $\mu_w(\mathbf{q}, \mathbf{x}_i)$ $(i = 1, \ldots, n)$ by $\psi(\mathbf{q})$, then $N(\mathbf{q})$ can be represented as $N(\mathbf{q}) = \{\mathbf{x}_l \mid \mu_w(\mathbf{q}, \mathbf{x}_l) \geq \psi(\mathbf{q})\}$.

In the test phase, instead of using the traditional majority voting scheme on the K neighbors, we use the following decision function to classify $\mathbf{q}$:

$$F(\mathbf{q}) = \arg\max_c V_c^K(\mathbf{q}), \tag{4}$$

where $V_c^K(\mathbf{q})$ represents the vote of class c obtained by using K nearest neighbors and it can be determined by

$$V_c^K(\mathbf{q}) = \sum_{\mathbf{x}_l \in N(\mathbf{q})} I(y_l = c)\ \mu_w(\mathbf{q}, \mathbf{x}_l), \tag{5}$$

where $I(.)$ denotes an indicator function, whose value is 1 if its argument is true and 0 otherwise.

3 Learning the Instance Weights by WDKNN Algorithm

The WDKNN algorithm presented in this section is an extension of the WDNN algorithm for $K \geq 1$. The WDKNN algorithm assigns a weight $w_i (\geq 0)$ to each training instance $\mathbf{x}_i$ $(i = 1, \ldots, n)$ by maximizing the LOOCV accuracy. This procedure is done in the training phase. Only the training instances and the associated weights with $w_i > 0$ are retained in the test phase.

The WDKNN algorithm is a hill climbing optimization technique for solving w_i, where the optimal weight $w_i^\star$ is determined by assuming all the other weights w_j $(j \neq i)$ are given and fixed. We set $w_i = 1$ $\forall i$ as the initial values. At iteration i $(i = 1, \ldots, n)$, the optimal weight $w_i^\star$ can be found by maximizing the objective function related to the LOOCV accuracy:

$$J(w_i) = \sum_{\{\mathbf{x}_m \in \Omega, \mathbf{x}_m \neq \mathbf{x}_i\}} I(F(\mathbf{x}_m | w_i) = y_m), \tag{6}$$

where $F(\mathbf{x}_m|w_i)$ is the decision function of instance $\mathbf{x}_m$ given that the weight for $\mathbf{x}_i$ is w_i. Here, $\mathbf{x}_m$ is treated as a query in the LOOCV procedure. Obviously, the obtained $w_i^\star$ is only suboptimal as the other weights change during the optimization process. Thus, the algorithm will be restarted after a fixed number of runs over the training data set. We follow [2] to restart the optimization process after three runs over the entire data in all experiments conducted in Section 4.

To optimize the weight w_i for the training instance $\mathbf{x}_i$, we assume that $\mathbf{x}_i$ is *consistent* on all other training instances $\mathbf{x}_m (m \neq i)$ in the LOOCV test. Below is the precise definition.

Definition: A training instance is *consistent* on a query if it can either make this query be classified into its class or it is irrelevant of its prediction.

In our proposal, the decision function $F(\mathbf{x}_m|w_i)$ of instance $\mathbf{x}_m$ is modeled as follows:

$$F'(\mathbf{x}_m|w_i) = I(w_i \leq \theta_m)\ F(\mathbf{x}_m|w_i = 0) + I(w_i > \theta_m)\ y_i, \tag{7}$$

where $F(\mathbf{x}_m|w_i = 0)$ is the decision function with $w_i = 0$, θ_m is a threshold for w_i. This model essentially means that $\mathbf{x}_m$ will be classified without using $(\mathbf{x}_i, w_i)$ unless they can make $\mathbf{x}_m$ be classified into its class y_i. It is easy to see that $F'(\mathbf{x}_m|w_i) = F(\mathbf{x}_m|w_i)$ for $K = 1$. This model is illustrated in Figure 1, where a three-class classification problem in two-dimensional space has been plotted. The class memberships of the training instances are distinguished by the various shapes of the data points. The query $\mathbf{q}$ is denoted by a black square and the instance $\mathbf{x}_i$ is the data point with a green '+' label. If we set $w_i = 0,\ w_j = 1, \forall j \neq i$, then the $K(= 7)$ nearest neighbors of $\mathbf{q}$ are drawn with connection lines and $\mathbf{q}$ will be classified into the 'triangle' class. On the other hand, if $w_i > 1.18,\ w_j = 1, \forall j \neq i$, the nearest neighbor with the dashed connection line will be replaced by $\mathbf{x}_i$ and thus WDKNN will assign $\mathbf{q}$ into the 'diamond' class. Hence, we have $\theta_m = 1.18$ in this example.

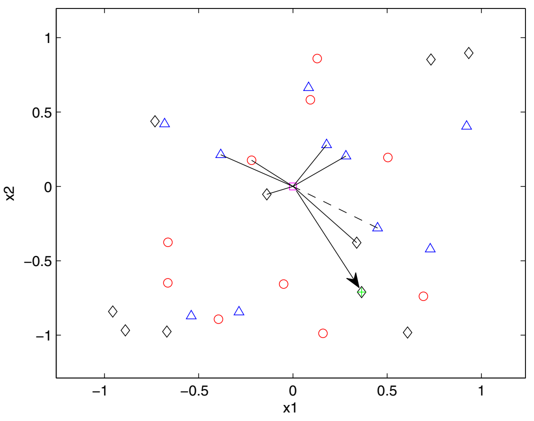


Fig. 1. An illustration of the model in equation (7)

The value of threshold θ_m can be specified for each $\mathbf{x}_m$ by considering three cases regarding the relationship between $\mathbf{x}_m$ and $(\mathbf{x}_i, w_i)$.

Case 1: If we have $F(\mathbf{x}_m|w_i = 0) = y_i$, then $w_i^\star$ will not depend on $(\mathbf{x}_m, y_m)$.

Proof. If $F(\mathbf{x}_m|w_i = 0) = y_i$, then according to the definition given in equation (4), we have[1]

$$V_{y_i}(\mathbf{x}_m|w_i = 0) > V_{c \neq y_i}(\mathbf{x}_m|w_i = 0). \tag{8}$$

But, according to equation (5), we also have

$$V_{y_i}(\mathbf{x}_m|w_i > 0) \geq V_{y_i}(\mathbf{x}_m|w_i = 0), \tag{9}$$
$$V_{c \neq y_i}(\mathbf{x}_m|w_i > 0) \leq V_{c \neq y_i}(\mathbf{x}_m|w_i = 0), \tag{10}$$

which leads to

$$V_{y_i}(\mathbf{x}_m|w_i > 0) > V_{c \neq y_i}(\mathbf{x}_m|w_i > 0), \tag{11}$$

and thus

$$F(\mathbf{x}_m|w_i > 0) = y_i. \tag{12}$$

Now,

$$F(\mathbf{x}_m|w_i > 0) = F(\mathbf{x}_m|w_i = 0), \tag{13}$$

which implies that $F(\mathbf{x}_m|w_i)$ and hence $I(F(\mathbf{x}_m|w_i) = y_m)$ will be a constant for all values of $w_i \geq 0$. Therefore, the optimal value of w_i is irrelevant of $(\mathbf{x}_m, y_m)$. (Note that this proof is valid regardless the appropriateness of the model given in equation (7).)

Case 2: If we have $F(\mathbf{x}_m|w_i = 0) \neq y_i$, $F(\mathbf{x}_m|w_i = 0) \neq y_m$ and $y_i \neq y_m$, then $w_i^\star$ will not depend on $(\mathbf{x}_m, y_m)$.

[1] For the sake of clarity, the notation $V_c^K(\mathbf{q})$ is replaced by $V_c(\mathbf{q})$ for the proof given here.

Proof. From the above given conditions, it is easy to arrive at

$$F'(\mathbf{x}_m|w_i) \neq y_m, \quad \forall w_i \geq 0, \tag{14}$$

leading to

$$I(F'(\mathbf{x}_m|w_i) = y_m) = 0, \quad \forall w_i \geq 0. \tag{15}$$

Hence, it can be seen that $\mathbf{x}_m$ will be misclassified for all values of w_i in this case.

Case 3: If we let M_i denote the collection of all training instances $\{(\mathbf{x}_m, y_m) \in \Omega \mid m \neq i\}$ that do not belong to Case 1 and 2. Then, the threshold value θ_m can be determined for all $(\mathbf{x}_m, y_m) \in M_i$ as follows:

$$\theta_m = \max\{\ \alpha_m,\ \beta_m\}, \tag{16}$$

$$\alpha_m = \psi(\mathbf{x}_m)/\mu(\mathbf{x}_m, \mathbf{x}_i), \tag{17}$$

$$\beta_m = (\max_c V_c^{K-1}(\mathbf{x}_m) - V_{y_i}^{K-1}(\mathbf{x}_m))/\mu(\mathbf{x}_m, \mathbf{x}_i). \tag{18}$$

where $\psi(\mathbf{x}_m)$ is the Kth largest value of $\mu_w(\mathbf{x}_m, \mathbf{x}_j)$ for $j = 1, \ldots, n$ and $j \neq i$.

Proof. Equation (7) tells us that $F'(\mathbf{x}_m|w_i) = y_i$ if and only if $w_i > \theta_m$. Also, according to the classification rule introduced in Section 2, $F(\mathbf{x}_m|w_i) = y_i$ if and only if two two conditions are satisfied: (a) $(\mathbf{x}_i, y_i)$ is selected as one of the K nearest neighbors of $\mathbf{x}_m$; (b) the class vote for class y_i is the largest among all classes.

In order to satisfy condition (a), we must have

$$w_i \cdot \mu(\mathbf{x}_m, \mathbf{x}_i) > \psi(\mathbf{x}_m), \tag{19}$$

leading to

$$w_i > \psi(\mathbf{x}_m)/\mu(\mathbf{x}_m, \mathbf{x}_i). \tag{20}$$

(b) After condition (a) is satisfied, the previous Kth neighbor of $\mathbf{x}_m$ will be replaced by $(\mathbf{x}_i, y_i)$. By equation (4) - (5),

$$w_i \cdot \mu(\mathbf{x}_m, \mathbf{x}_i) + V_{y_i}^{K-1}(\mathbf{x}_m) > V_c^{K-1}(\mathbf{x}_m) \quad \forall\ c = 1, \ldots, C, \tag{21}$$

which leads to

$$w_i > (\max_c V_c^{K-1}(\mathbf{x}_m) - V_{y_i}^{K-1}(\mathbf{x}_m))/\mu(\mathbf{x}_m, \mathbf{x}_i). \tag{22}$$

According to the above three cases, the optimal $w_i^\star$ can be found based on all $\mathbf{x}_m \in M_i$ by maximizing the following criterion:

$$J'(w_i) = \sum_{\{\mathbf{x}_m \in M_i\}} I(F'(\mathbf{x}_m|w_i) = y_m). \tag{23}$$

Assume there are L threshold values θ_m ranked in ascending order $\theta_1 < \theta_2 < \ldots < \theta_L$. Then, $L+1$ values of w_i are examined and the best one can be found by using equation (23). The first and last values examined are 0 and $\theta_L + \pi$, respectively (π is a very small positive number). The rest values are chosen in the middle of two successive θ_m.

The compression rate (CR) is used to compute the rate at which the prototypes are reduced for a prototype reduction method:

$$CR = 1 - r/n, \tag{24}$$

where r and n represent the reduced and original number of instances, respectively. The KNN (and hence NN) classifier retains all training instances and thus it has $CR = 0$.

4 Experiments

To validate the proposed WDKNN algorithm, we compared it with the traditional KNN (including NN) algorithm and several other state-of-the-art prototype reduction methods in literature: learning vector quantization [4], learning prototypes and distances (LPD) [6], adaptive nearest neighbor (A-NN) [11], prototype-dependent weighting (PW) [7], WDNN [2] and MWDNN [2]. Twenty real world benchmark data sets taken from the UCI Machine Learning Repository[2] are used throughout the experiments (see Table 1). We performed the implementation using MATLAB R2007a on Windows XP with 2Duo CPU running on 3.16 GHz PC with 3.25 GB RAM.

4.1 Experiments on UCI Data Sets

We follow [7] [6] and [2] by using five-fold cross-validation (5-CV) to test the performance of various methods. The average accuracy and compression rate for various methods are compared. Both the KNN and WDKNN algorithms have a tuning parameter: K (neighborhood size). They are determined by selecting the integer ranging from 1 to 41 that corresponds to the maximal 5-CV accuracy on each data set. The selected values of K for both algorithms on each data set are reported in Table 1.

Figure 2 shows the LOOCV accuracies of WDNN and WDKNN during the weight learning progress at the first iteration. This figure was plotted based on one fold of a 5-CV test on the ionosphere data set and $K = 4$ was used for WDKNN. The LOOCV accuracy was computed after each one of the weights had been updated. It can be seen that the LOOCV accuracy for WDKNN never increases during the learning process. Also, WDKNN has shown superior performance to WDNN during the whole optimization process.

The mean classification accuracies and compression rates of the NN, KNN, WDNN and WDKNN methods are summarized in Table 1. We noted that the accuracy of WDKNN is higher than that of WDNN over all data sets. However, this higher accuracy is achieved at the expense of lower compression rates. This

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

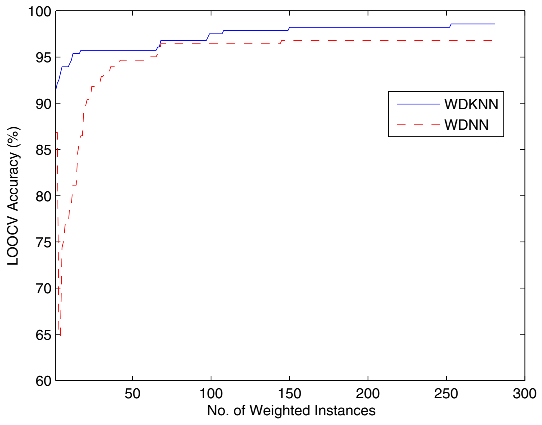


Fig. 2. Leave-one-out cross-validation accuracy during the first iteration of the WDNN and WDKNN ($K = 4$) algorithms on ionosphere data

Table 1. Classification accuracy (%) and compression rates (%) of four classifiers on twenty real data sets

Data	NN	KNN	CR	WDNN	CR	WDKNN	CR	K_{knn}	K_{wdknn}
Australian	79.28	86.52	0	82.75	91.88	**86.67**	79.67	33	24
Balance	80.48	90.08	0	86.72	93.32	**91.36**	70.96	20	36
Breast (Original)	95.90	97.07	0	96.63	98.72	**97.22**	96.82	5	7
Breast (Prognostic)	69.21	78.79	0	74.79	93.57	**79.81**	63.26	6	31
Dermatology	94.54	96.18	0	95.91	94.06	**97.00**	83.20	10	21
Diabetes	71.75	**76.43**	0	71.87	89.55	75.65	56.48	18	40
Ecoli	82.73	86.59	0	83.32	93.30	**88.08**	84.52	9	8
Flag	41.30	46.98	0	43.86	79.13	**51.08**	70.88	28	3
German	68.30	73.50	0	69.00	88.95	**74.50**	60.25	22	33
Glass	**73.39**	**73.39**	0	64.41	81.19	73.32	70.68	1	3
Haberman	65.34	74.53	0	70.93	94.28	**76.17**	71.41	11	29
Heart	78.15	84.07	0	82.22	91.02	**84.81**	83.61	29	3
Ionosphere	87.45	87.45	0	90.31	93.59	**93.16**	88.60	1	4
Iris	92.67	96.67	0	94.00	91.00	**97.33**	59.00	13	33
Liver	60.87	69.86	0	66.96	89.28	**70.43**	52.03	7	25
Soybean	95.56	95.56	0	84.67	73.93	**97.78**	55.87	1	7
Vehicle	70.69	**73.88**	0	63.31	84.43	71.52	53.99	5	29
Vote	93.11	93.11	0	90.95	94.72	**95.28**	84.16	1	14
Wine	95.51	97.75	0	95.49	93.96	**98.30**	82.86	33	12
Zoo	94.00	**96.00**	0	95.00	82.43	**96.00**	76.74	6	4
Average	79.51	83.72	0	80.16	89.62	**84.77**	72.25	13	18

The bold numbers represent the highest accuracy values for each data set.

is because setting a larger K usually requires more training instances to be involved in the classification stage. It can be seen that WDKNN obtained the overall highest accuracy and also achieved the best accuracy in 17 out of 20

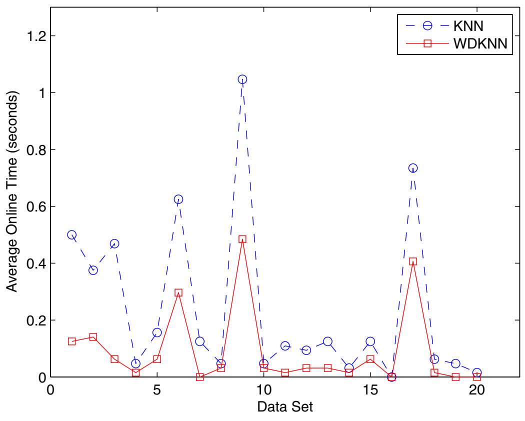


Fig. 3. The comparison of average online time (in seconds) between KNN and WDKNN over the real data sets

cases (with 1 equal best one). Based on the accuracy values obtained by various methods over twenty data sets, we used a one-tailed paired t-test to compare WDKNN's accuracies to those of NN, KNN and WDNN. We have statistical evidence that the average accuracy of WDKNN is significantly higher than the average accuracies of all of NN, KNN and WDNN for each significance level of $0.5\%, 1\%$ and 5%. Moreover, WDKNN achieves higher accuracy than KNN with no more than 30% of the training data set on average.

Figure 3 shows the average online time required for KNN and WDKNN over the twenty real data sets. In the figure, each data set was denoted by the order in which they appeared in Table 1. We noted that the online time required for WDKNN is less than that of KNN over all 20 data sets. The online speed of WDKNN is 7.5 times faster than KNN for the breast cancer (original) data set, and the average online speed of WDKNN over the twenty data sets is 2.6 times faster than that of KNN. Although the reduction in online time of KNN is trivial for these small to medium sized data sets, the reduction can be dramatic for large data sets and this property is crucial for a KNN classifier. If the size of either the training set or the test set increases, the advantage of WDKNN over KNN in terms of online execution speed will become more obvious.

Table 2 gives the average 5-CV accuracies of WDKNN in comparison with the published results of several state-of-the-art prototype reduction methods. The data sets used by these methods are taken from the Statlog project, so we only display WDKNN's results on these data sets. The experimental procedures for the competing methods are the same as those of WDKNN except that the feature weighting methods have only been used in PW, LPD, MDNN and MWDNN. In this paper, we focus on instance weighting, and thus we only employ the basic Euclidean distance, which implicitly assumes that all features have equal weight. Hence, the accuracies of WDKNN would be consistently higher if the feature weighting scheme had also been used in WDKNN. Despite this unfairness for WDKNN, it still achieves the best average accuracy on all data sets.

Table 2. Classification accuracies (%) of the WDKNN algorithm in comparison with other algorithms in literature

Data	WDKNN	LVQ1	LVQ2	LVQ3	A-NN	PW	LPD	WDNN	MWDNN
Australian	**86.67**	66.7	66.0	68.9	75.91	83.05	86.1	85.48	85.01
Balance	**91.36**	83.7	85.3	83.7	89.88	86.56	83.7	89.14	90.32
Breast (Original)	97.22	95.6	95.5	95.8	97.14	96.68	96.6	97.52	**97.88**
Diabetes	75.65	73.6	74.2	74.0	71.86	72.61	74.0	75.96	**76.31**
German	74.50	69.9	71.5	71.3	61.89	71.68	74.0	**75.84**	73.89
Glass	73.32	60.4	57.6	58.5	71.22	73.72	72.0	71.34	70.81
Heart	84.81	64.0	66.0	66.0	67.45	81.06	81.4	83.91	**84.91**
Liver	**70.43**	67.5	67.3	66.4	65.12	63.78	66.7	68.31	65.39
Vehicle	71.52	61.8	68.0	65.6	66.28	70.69	**72.6**	70.14	69.15
Vote	**95.28**	92.4	93.8	94.3	93.31	94.49	96.3	92.29	91.37
Wine	**98.30**	70.3	74.2	70.8	84.82	98.65	95.0	96.61	96.04
Average	**83.6**	73.3	74.5	74.1	76.8	81.2	81.7	82.4	81.9

The bold numbers represent the highest accuracy values for each data set.

Using the one-tailed paired t-test (with a significance level of 5%), we have statistical evidence that WDKNN outperforms all the competing methods. It must be noted that the A-NN and PW methods do not reduce the training size. In addition, these results obtained by WDKNN are comparable to or better than those obtained by other state-of-the-art methods recently published on the same tasks [9,8,5].

4.2 Effect of Noise

Since WDKNN is designed to be more robust in the presence of noise than WDNN, the experiments conducted in Section 4.1 were repeated with 20% uniform class noise artificially added to each data set. This was done by randomly changing the class label of 20% of the training instances to an incorrect value (with an equal probability for each of the incorrect classes). The class labels of the test instances are not noisy. Note that the experimental settings here are the same as those in [2]. The performance of NN, KNN, WDNN and WDKNN were tested to see their robustness of noise.

Table 3 reports the average accuracies and compression rates of each method over twenty data sets. As can be seen, the accuracy of WDKNN is much better than that of WDNN. This is consistent with our motivation that using $K > 1$ can reduce WDNN's sensitivity of noise. The average values of K for both KNN and WDKNN on the noisy data become larger compared to the real data sets. This also suggests that a larger K is more suitable for the noisy data sets. WDKNN achieves the highest accuracy value averaged over the entire data set. Also, the accuracy given by WDKNN is the best in 10 out of 20 cases. Using the one-tailed paired t-test (for both the 1% and 5% significance level), we have statistical evidence that WDKNN outperforms NN and WDNN in terms of accuracy, and we have no statistical evidence that the average accuracy of WDKN is larger

Table 3. Classification accuracies (%) and compression rates (%) of four classifiers on the noisy data sets

Data	NN	KNN	CR	WDNN	CR	WDKNN	CR	K_{knn}	K_{wdknn}
Australian	66.67	85.51	0	77.25	88.99	**85.94**	67.07	39	31
Balance	69.60	**88.96**	0	80.96	90.16	88.00	62.36	23	38
Breast (Original)	80.09	**96.48**	0	93.56	89.89	96.05	89.42	28	33
Breast (Prognostic)	64.69	**78.33**	0	69.24	89.77	78.31	66.55	10	14
Dermatology	74.07	**95.36**	0	91.27	91.60	95.08	74.59	11	36
Diabetes	62.11	**73.17**	0	66.67	86.72	72.14	60.41	36	18
Ecoli	65.17	84.51	0	79.74	91.52	**84.81**	69.79	14	39
Flag	36.65	47.49	0	42.31	81.19	**50.05**	64.17	33	11
German	63.60	72.60	0	64.90	88.03	**73.20**	67.13	30	18
Glass	62.20	65.49	0	64.50	82.36	**67.32**	76.52	4	1
Haberman	60.15	**74.53**	0	71.59	92.81	73.55	73.29	23	17
Heart	64.81	**84.44**	0	72.96	86.76	83.70	59.26	26	38
Ionosphere	77.77	85.17	0	85.45	89.74	**88.89**	71.37	10	12
Iris	69.33	95.33	0	88.67	88.67	**96.67**	66.00	13	25
Liver	55.07	**66.09**	0	58.55	85.51	65.80	41.88	26	35
Soybean	80.89	93.33	0	86.89	75.48	**93.56**	49.43	4	11
Vehicle	57.09	**69.98**	0	61.59	84.49	67.49	56.74	15	26
Vote	71.96	93.11	0	85.39	89.22	**93.55**	65.20	16	38
Wine	80.37	**97.17**	0	92.13	91.15	96.63	68.26	37	29
Zoo	71.14	93.05	0	86.14	78.97	**93.10**	73.27	5	6
Average	66.67	82.01	0	75.99	87.15	**82.19**	66.14	20	24

The bold numbers represent the highest accuracy values for each data set.

than that of KNN. On average, WDKNN achieves the same level of accuracy as KNN by using no more than 35% of the training data set. In addition, we also found that the online time required for WDKNN is less than that of KNN over all noisy data sets. Specifically, the average online speed of WDKNN over the twenty noisy data sets is 2.2 times faster than that of KNN.

5 Conclusions

In this paper, a novel prototype reduction method for KNN has been proposed. This method removes the instances that are more of a computational burden but do not contribute to better classification accuracy and it also assigns a weight to each retained training instance. These weights are learned by maximizing the leave-one-out cross-validation classification accuracy, which is the true estimate of the generalization ability of a classifier. Empirical results have shown that WDKNN considerably reduces the size of the training set while retaining or improving the classification accuracy of KNN. In fact, the proposed method may also be useful for other lazy learning methods such as ALH [10] in terms of storage requirement and online speed.

References

1. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
2. Jahromi, M.Z., Parvinnia, E., John, R.: A method of learning weighted similarity function to improve the performance of nearest neighbor. Information Sciences 179(17), 2964–2973 (2009)
3. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning, 2nd edn. Springer, Heidelberg (2009)
4. Kohonen, T.: Self-Organization and Associative Memory, 3rd edn. Springer, Berlin (1989)
5. Loog, M., Duin, R.P.W.: Linear dimensionality reduction via a heteroscedastic extension of LDA: the chernoff criterion. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(6), 732–739 (2004)
6. Paredes, R., Vidal, E.: Learning prototypes and distances: a prototype reduction technique based on nearest neighbor error minimization. Pattern Recognition 39(2), 180–188 (2006)
7. Paredes, R., Vidal, E.: Learning weighted metrics to minimize nearest-neighbor classification error. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(7), 1100–1110 (2006)
8. Peng, J., Heisterkamp, D.R., Dai, H.: Adaptive quasiconformal kernel nearest neighbor classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(5), 656–661 (2004)
9. Ridder, D., Loog, M., Reinders, M.J.T.: Local Fisher embedding. In: Proc. of 17th International Conference on Pattern Recognition, vol. 2, pp. 295–298 (2004)
10. Yang, T., Kecman, V.: Adaptive local hyperplane classification. Neurocomputing 71, 3001–3004 (2008)
11. Wang, J., Neskovic, P., Cooper, L.N.: Improving nearest neighbor rule with a simple adaptive distance measure. Pattern Recognition Letters 28(2), 207–213 (2007)
12. Wilson, D.R., Martinez, T.R.: Reduction techniques for exemplar-based learning algorithms. Machine Learning 38(3), 257–286 (2000)

Generalized Two-Dimensional FLD Method for Feature Extraction: An Application to Face Recognition

Shiladitya Chowdhury[1], Jamuna Kanta Sing[2], Dipak Kumar Basu[2], and Mita Nasipuri[2]

[1] Department of Master of Computer Application, Techno India, Kolkata, India
[2] Department of Computer Science & Engineering, Jadavpur University, Kolkata, India
dityashila@yahoo.com, jksing@ieee.org, dipakkbasu@gmail.com, mitanasipuri@yahoo.com

Abstract. This paper presents a novel scheme for face feature extraction, namely, the generalized two-dimensional Fisher's linear discriminant (G-2DFLD) method. The G-2DFLD method is an extension of the 2DFLD method for feature extraction. Like 2DFLD method, G-2DFLD method is also based on the original 2D image matrix. However, unlike 2DFLD method, which maximizes class separability either from row or column direction, the G-2DFLD method maximizes class separability from both the row and column directions simultaneously. In G-2DFLD method, two alternative Fisher's criteria have been defined corresponding to row and column-wise projection directions. The principal components extracted from an image matrix in 2DFLD method are vectors; whereas, in G-2DFLD method these are scalars. Therefore, the size of the resultant image feature matrix is much smaller using G-2DFLD method than that of using 2DFLD method. The proposed G-2DFLD method was evaluated on two popular face recognition databases, the AT&T (formerly ORL) and the UMIST face databases. The experimental results show that the new G-2DFLD scheme outperforms the PCA, 2DPCA, FLD and 2DFLD schemes, not only in terms of computation times, but also for the task of face recognition using a multi-class support vector machine.

Keywords: Generalized two-dimensional FLD, Feature extraction, Face recognition.

1 Introduction

The Fisher's linear discriminant (FLD) method has been widely used for feature extraction and dimension reduction in pattern recognition and computer vision domains. The key idea of the FLD technique is to find the optimal projection that maximizes the ratio of the *between-class* and the *within-class* scatter matrices of the projected samples. A difficulty in using the FLD technique in face recognition is the "*small sample size (SSS)*" problem [1]. This problem usually arises when the number of samples is smaller than the dimension of the samples. In face recognition domain, the dimension of a face image is generally very high. Therefore, the within-class scatter matrix is almost always singular, thereby making the implementation of FLD method impossible. One direct solution of the *SSS* is to down sample the face images into a considerably small size and then perform FLD technique. However, this process is not computationally efficient as the pre-processing of images takes considerable amount of time before actual application of the FLD technique. Er *et al.* [2] proposed a PCA+FLD technique to avoid the *SSS* problem. In [2], face features are first extracted by the principal component analysis (PCA) method and then the resultant features are further processed by the FLD technique to acquire lower-dimensional discriminant features. An improved PCA technique, the two-dimensional PCA (2DPCA), was proposed by Yang *et al.* [3]. Unlike PCA, which works on the stretched image vector, the 2DPCA works directly on the original 2D image matrix. The 2DPCA is not only computationally efficient, but also superior for the task of face recognition and image reconstruction than the conventional PCA technique [3]. However, the PCA techniques yield projection directions that maximize the total scatter across all classes, i.e., across all face images. Therefore, the PCA retains unwanted variations caused by lighting, facial expression, and other factors [2], [4]. The PCA techniques do not provide any information for class discrimination but dimension reduction [2]. Recently, Xiong *et al.* [5] proposed a two-dimensional FLD (2DFLD) method, which also works directly on the original 2D image matrix and maximizes class separability either from row or column direction. The so called *SSS* problem does not arise in 2DFLD method as the size of its scatter matrices is much smaller. The 2DFLD method is found to be superior to the PCA and 2DPCA in terms of feature extraction and face recognition [5].

In this paper, we have extended the 2DFLD algorithm [5] and present a novel generalized two-dimensional FLD (G-2DFLD) technique, which maximizes class separability from both the row and column directions simultaneously. Like 2DFLD method, G-2DFLD method is also based on the original 2D image matrix. In G-2DFLD method, two alternative Fisher's criteria have been defined corresponding to row and column-wise projection directions. Unlike 2DFLD method, the principal components extracted from an image matrix by the G-2DFLD method are scalars. Therefore, the size of the resultant image feature matrix is much smaller using the G-2DFLD method than that of using the 2DFLD method. The experimental results on the AT&T and the UMIST databases show that the new G-2DFLD scheme outperforms the PCA, 2DPCA, FLD and 2DFLD schemes, not only in terms of computation time, but also for the task of face recognition using a multi-class support vector machine (SVM).

The remaining part of the paper is organized as follows. Section 2 describes the procedure of extracting face features using 2DFLD technique. Section 3 presents the key idea and algorithm of the proposed G-2DFLD method for feature extraction. The experimental results on the AT&T and the UMIST face databases are presented in Section 4. Finally, Section 5 draws the conclusion remarks.

2 Two-Dimensional FLD (2DFLD) Method for Feature Extraction

The 2DFLD [5] method is based on the 2D image matrix. It does not need to form a stretched large image vector from the 2D image matrix. The key idea is to project an image matrix **X**, an $m\times n$ random matrix, onto a projection matrix **A** of dimension $n\times k$ *($k\leq n$)* to get an image feature matrix **Y** of dimension $m\times k$ by the following linear transformation [5]:

$$\boldsymbol{Y} = \boldsymbol{XA} \tag{1}$$

Let there are N training images, each one is denoted by $m\times n$ image matrix $\mathbf{X}_i$ (i=1, 2, …, N). The training images contain C classes (subjects), and the cth class C_c has N_c samples ($\sum_{c=1}^{C} N_c = N$). Let the mean image of the training samples is denoted by $\boldsymbol{\mu}$ and the mean image of the cth class is denoted by $\boldsymbol{\mu}_c$. The between-class and within-class scatter matrices $\boldsymbol{G}_b$ and $\boldsymbol{G}_w$, respectively are defined as follows:

$$\mathrm{G_b} = \sum_{c}^{C} \mathrm{N_c} (\mu_c - \mu)^T (\mu_c - \mu) \tag{2}$$

$$\mathrm{G_w} = \sum_{c}^{C} \sum_{i\in c}^{N} (\mathrm{X}_i - \mu_c)^T (\mathrm{X}_i - \mu_c) \tag{3}$$

Then the two-dimensional Fisher's criterion J(Q) is defined as follows:

$$J(Q) = \frac{\left|Q^T G_b Q\right|}{\left|Q^T G_w Q\right|} \tag{4}$$

where Q is the projection matrix.

It may be noted that the size of both the $\boldsymbol{G}_b$ and $\boldsymbol{G}_w$ is $n\times n$. If $\boldsymbol{G}_w$ is a nonsingular matrix, the ratio in (4) is maximized when the column vectors of the projection matrix **Q**, are the eigenvectors of $G_b G_w^{-1}$. The optimal projection matrix $\mathbf{Q}_{\mathrm{opt}}$ is defined as follows:

$$\begin{aligned} Q_{opt} &= \arg\max_{Q} \left|G_b G_w^{-1}\right| \\ &= [q_1, q_2, \ldots, q_k] \end{aligned} \tag{5}$$

where $\{q_i \mid i=1, 2, \ldots, k\}$ is the set of normalized eigenvectors of $G_b G_w^{-1}$ corresponding to k largest eigenvalues $\{\lambda_i \mid i=1, 2, \ldots, k\}$.

Now, each face image $\mathbf{X}_i$ (i=1, 2, …, N) is projected into the optimal projection matrix $\mathbf{Q}_{\mathrm{opt}}$ to obtain its $(m\times k)$-dimensional 2DFLD-based features $\mathbf{Y}_i$, which is defined as follows:

$$Y_i = \overline{X}_i Q_{opt} \;; i = 1,2,\ldots, N \tag{6}$$

where $\overline{X_i}$ is mean-subtracted image of $\mathbf{X}_i$

3 Generalized Two-Dimensional FLD (G-2DFLD) Method for Feature Extraction

3.1 Key Idea and the Algorithm

Like 2DFLD method, the generalized two-dimensional FLD (G-2DFLD) method is also based on 2D image matrix. The only difference is that, it maximizes class separability from both the row and column directions simultaneously by the following linear transformation:

$$\boldsymbol{Z} = \boldsymbol{U}^{T} \boldsymbol{X} \boldsymbol{V} \tag{7}$$

where **U** and **V** are two projection matrices of dimension $m \times p$ $(p \leq m)$ and $n \times q$ $(q \leq n)$, respectively. Therefore, our goal is to find the optimal projection directions **U** and **V** so that the projected vector in the $(p \times q)$-dimensional space reaches its maximum class separability.

3.1.1 Alternate Fisher's Criteria

We have defined two alternative Fisher's criteria $J(U)$ and $J(V)$ corresponding to row and column-wise projection directions as follows:

$$J(U) = \frac{\left| U^{T} G_{br} U \right|}{\left| U^{T} G_{wr} U \right|} \tag{8}$$

and

$$J(V) = \frac{\left| V^{T} G_{bc} V \right|}{\left| V^{T} G_{wc} V \right|} \tag{9}$$

where

$$\mathrm{G}_{\mathrm{br}} = \sum_{\mathrm{c}}^{\mathrm{C}} \mathrm{N}_{\mathrm{c}} (\mu_{\mathrm{c}} - \mu)(\mu_{\mathrm{c}} - \mu)^{\mathrm{T}} \tag{10}$$

$$\mathrm{G}_{\mathrm{wr}} = \sum_{\mathrm{c}}^{\mathrm{C}} \sum_{\mathrm{i} \in \mathrm{c}}^{\mathrm{N}} (\mathrm{X}_{\mathrm{i}} - \mu_{\mathrm{c}})(\mathrm{X}_{\mathrm{i}} - \mu_{\mathrm{c}})^{\mathrm{T}} \tag{11}$$

$$\mathrm{G}_{\mathrm{bc}} = \sum_{\mathrm{c}}^{\mathrm{C}} \mathrm{N}_{\mathrm{c}} (\mu_{\mathrm{c}} - \mu)^{\mathrm{T}} (\mu_{\mathrm{c}} - \mu) \tag{12}$$

$$\mathrm{G}_{\mathrm{wc}} = \sum_{\mathrm{c}}^{\mathrm{C}} \sum_{\mathrm{i} \in \mathrm{c}}^{\mathrm{N}} (\mathrm{X}_{\mathrm{i}} - \mu_{\mathrm{c}})^{\mathrm{T}} (\mathrm{X}_{\mathrm{i}} - \mu_{\mathrm{c}}) \tag{13}$$

We call the matrices $\boldsymbol{G}_{br}$, $\boldsymbol{G}_{wr}$, $\boldsymbol{G}_{bc}$ and $\boldsymbol{G}_{wc}$, as *image row between-class scatter matrix*, *image row within-class scatter matrix*, *image column between-class scatter matrix* and *image column within-class scatter matrix*, respectively. It may be noted that size of the scatter matrices $\boldsymbol{G}_{br}$ and $\boldsymbol{G}_{wr}$ is $m\times m$, whereas, for $\boldsymbol{G}_{bc}$ and $\boldsymbol{G}_{wc}$ the size is $n\times n$. The sizes of these scatter matrices are much smaller than that of the conventional FLD algorithm, whose scatter matrices are $mn\times mn$ in size. For a square image, $m=n$ and we have $\mathbf{G}_{\mathrm{br}} = G_{bc}^T$ and $\mathbf{G}_{\mathrm{wr}} = G_{wc}^T$ and vice-versa.

The ratios in (8) and (9) are maximized when the column vectors of the projection matrix **U** and **V**, are the eigenvectors of $G_{br}G_{wr}^{-1}$ and $G_{bc}G_{wc}^{-1}$, respectively. The optimal projection (eigenvector) matrix $\mathbf{U}_{opt}$ and $\mathbf{V}_{opt}$ are defined as follows:

$$\begin{aligned} U_{opt} &= \arg\max_{U} \left| G_{br} G_{wr}^{-1} \right| \\ &= [\mathrm{u}_1, \mathrm{u}_2, \ldots, \mathrm{u}_p] \end{aligned} \tag{14}$$

$$\begin{aligned} V_{opt} &= \arg\max_{V} \left| G_{bc} G_{wc}^{-1} \right| \\ &= [\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_q] \end{aligned} \tag{15}$$

where $\{\mathrm{u}_i \mid i=1, 2, \ldots, p\}$ is the set of normalized eigenvectors of $G_{br}G_{wr}^{-1}$ corresponding to p largest eigenvalues $\{\lambda_i \mid i=1, 2, \ldots, p\}$ and $\{\mathrm{v}_j \mid j=1, 2, \ldots, q\}$ is the set of normalized eigenvectors of $G_{bc}G_{wc}^{-1}$ corresponding to q largest eigenvalues $\{\alpha_j \mid j=1, 2, \ldots, q\}$.

3.1.2 Feature Extraction

The optimal projection matrices $\mathbf{U}_{\mathrm{opt}}$ and $\mathbf{V}_{\mathrm{opt}}$ are used for feature extraction. For a given image sample **X**, an image feature is obtained by the following linear projection:

$$z_{ij} = \boldsymbol{u}_i^{\boldsymbol{T}} \boldsymbol{X}\boldsymbol{v}_j, i=1,2,\ldots,p; j=1,2,\ldots,q \tag{16}$$

The z_{ij} (i=1, 2, …, p; j=1, 2, …, q) is called a *principal component* of the sample image **X**. It should be noted that each *principal component* of the 2DFLD method is a vector, whereas, the *principal component* of the G-2DFLD method is a scalar. The principal components thus obtained are used to form a G-2DFLD-based *image feature matrix* **Z** of dimension $p\times q$ $(p\leq m, q\leq n)$, which is much smaller than the 2DFLD-based *image feature matrix* **Y** of dimension $m\times k$ $(k\leq n)$. Therefore, in this case an image matrix is reduced considerably in both the row and column directions simultaneously.

4 Experimental Results

The performance of the proposed method has been evaluated on the AT&T Laboratories Cambridge database (formerly ORL database) [6] and the UMIST face database [7]. The AT&T database is used to test performance of the proposed method under the condition of minor variations of rotation and scaling and the UMIST database is used to examine the performance of the method when the angle of rotation of the facial images is quite large. The experiments were carried out in two different strategies; randomly partitioning the database and n-fold cross validation test.

We have designed a multi-class support vector machine (SVM) using Gaussian kernels for classification of the images to test the effectiveness of the G-2DFLD algorithm. The SVM has been recently proposed by Vapnik *et al.* [8] for binary classification and found to be very effective for pattern recognition. A SVM finds the hyperplane that separates the samples of two classes while maximizing the distance from either class to the hyperplane. This hyperplane is called Optimal Separating Hyperplane (OSH), which minimizes the risk of misclassification of both the training and test samples. A multi-class SVM has been designed by combining two class SVMs. In particular, we have adopted the *one-against-all* strategy to classify samples between each class and all the remaining classes. The one-against-all strategy is discussed as follows:

Let the training set ($\mathbf{X_i}$, c_i) consists of N samples of M classes, where c_i $(c_i \in 1,2,\ldots,M)$ represents the class label of the sample $\mathbf{X}_i$. An SVM is constructed for each class by discriminating that class from the remaining (M-1) classes. Thus the number of SVMs used in this approach is M. A test pattern $\mathbf{X}$ is classified by using the *winner-takes-all* decision strategy, i.e., the class with the maximum value of the discriminant function f($\mathbf{X}$) is assigned to it. All the N training samples are used in constructing an SVM for a class. The SVM for class k is constructed using the set of training samples and their desired outputs, ($\mathbf{X_i}$, y_i). The desired output y_i for a training sample $\mathbf{X}_i$ is defined as follows:

$$y_i = \begin{cases} +1 & \text{if} \quad c_i = k \\ -1 & \text{if} \quad c_i \neq k \end{cases} \tag{17}$$

The samples with the desired output y_i = +1 are called *positive* samples and the samples with the desired output y_i = -1 are called *negative* samples.

4.1 Experiments on the AT and T Face Database

The AT&T database contains 400 gray-scale images of 40 persons. Each person has 10 gray-scale images, having a resolution of 112×92 pixels. Images of the individuals have been taken by varying light intensity, facial expressions (open/closed eyes, smiling/not smiling) and facial details (glasses/no glasses) and against a dark homogeneous background, with tilt and rotation up to 20^o and scale variation up to 10%. Sample face images of a person are shown in Fig. 1.

4.1.1 Randomly Partitioning the Database

In this experimental strategy, we randomly select d images from each subject to form the training set and the remaining images are included in the test set. To ensure sufficient training and to test the effectiveness of the proposed technique for different sizes of the training sets, we choose the value of d as 3, 4, 5, 6 and 7. It should be noted that there is no overlap between the training and test images. To reduce the influence of performance on the training and test sets, for each value of d, experiment is repeated 20 times with different training and test sets. Since the numbers of projection vectors p and q have a considerable impact on the performance of the G-2DFLD algorithm, we perform several experiments by varying the values of p and q.

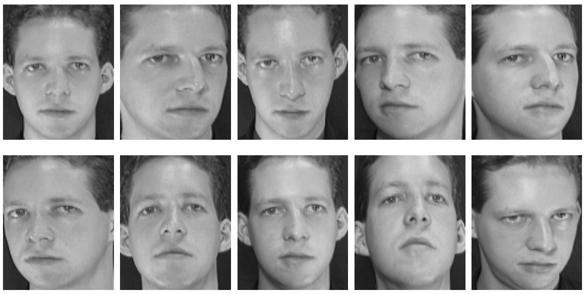


Fig. 1. Sample images of a subject from the AT&T database

Fig. 2 shows the recognition rates of the G-2DFLD algorithm using a multi-class support vector machine (SVM). For each value *d*, average recognition rates are plotted by varying the values of *p* and *q*. For *d*=3, 4, 5, 6 and 7 the best average recognition rates are found to be 92.82%, 95.94%, 97.68%, 98.72% and 98.42%, respectively and the dimension *(p*×*q)* of the corresponding image feature matrices are (16×16), (16×16), (14×14), (14×14) and (8×8), respectively.

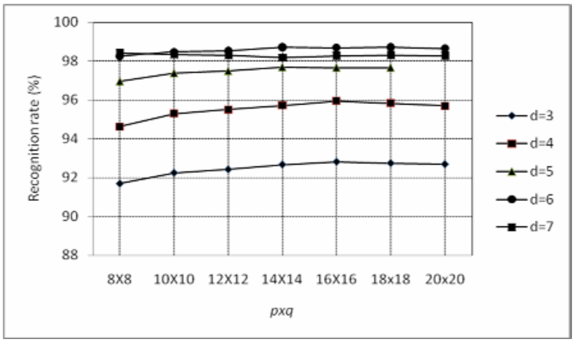


Fig. 2. Average recognition rates of the G-2DFLD algorithm on the AT&T database for different values *d* by varying the values of *p* and *q*

4.1.2 N-Fold Cross Validation Test

In this experiment, we divide the AT&T database (formerly ORL database) into ten-folds randomly, taking one image of a person into a fold. Therefore, each fold consists of 40 images, each one corresponding to a different person. For ten-folds cross validation test, in each experimental run, nine folds are used to train the multi-class SVM and remaining one fold for testing. Therefore, training and test sets consist of 360 and 40 images, respectively. The average recognition rates by varying the image feature matrix (i.e. *p*×*q*) are shown in Fig. 3. The best average recognition rate is found to be 99.75% using image feature matrix of size (8×8).

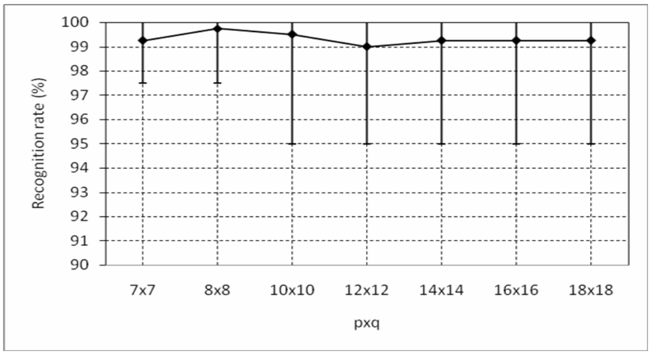


Fig. 3. Average recognition rates of the G-2DFLD algorithm on the AT&T database for 10-folds cross validation test by varying the values of p and q. The upper and lower extrema of the error bars represent the maximum and minimum values, respectively.

4.1.3 Comparison with Other Methods

For a fair comparison, we have implemented the PCA, 2DPCA, PCA+FLD and 2DFLD algorithms and used the same multi-class SVM and parameters for classification. The comparisons of the best average recognition rates of the PCA, 2DPCA, PCA+FLD and 2DFLD algorithms along with the proposed G-2DFLD algorithm using the two different experimental strategies are shown in Table 1. It may be noted that in all the cases the performance of the G-2DFLD method is better than the PCA, 2DPCA, PCA+FLD and 2DFLD methods.

Table 1. Comparison of different methods in terms of average recognition rates (%) on the AT&T database. Figures within the parentheses denote the number of features.

Experiment	Method	d=3	d=4	d=5	d=6	d=7
Randomly partition, d images/subject	G-2DFLD	92.82 (16×16)	95.94 (16×16)	97.68 (14×14)	98.72 (14×14)	98.42 (8×8)
	PCA	85.58 (60)	89.42 (60)	93.10 (60)	95.28 (60)	96.01 (60)
	2DPCA	91.27 (112×16)	94.33 (112×16)	96.83 (112×14)	97.72 (112×14)	97.79 (112×8)
	PCA+FLD	83.65 (25)	88.65 (25)	92.60 (25)	95.30 (25)	95.83 (25)
	2DFLD	92.30 (112×16)	95.08 (112×16)	97.50 (112×14)	98.26 (112×14)	97.88 (112×8)
10-folds cross validation test	G-2DFLD	99.75 (8×8)				
	PCA	97.00 (60)				
	2DPCA	99.25 (112×8)				
	PCA+FLD	98.25 (25)				
	2DFLD	99.00 (112×8)				

Table 2. Comparison of average feature extraction time (in seconds) using 200 training and 200 test images on the AT&T database

Method	# of features	Time (seconds)
G-2DFLD	14×14 = 196	12.95
PCA	60	55.10
2DPCA	112×14 = 1568	32.55
PCA+FLD	25	55.75
2DFLD	112×14 = 1568	22.35

Table 2 shows the average time (in seconds) taken by the G-2DFLD, PCA, 2DPCA, PCA+FLD and 2DFLD methods for feature extraction on the AT&T database using an IBM Intel Pentium 4 Hyper-Threading technology, 3.0 GHz, 2 GB DDR-II RAM computer running on Fedora 9 Linux Operating Systems. It may be again noted that the proposed G-2DFLD method is more efficient than the PCA, 2DPCA, PCA+FLD and 2DFLD methods in term of computation time.

4.2 Experiments on the UMIST Face Database

The UMIST[1] face database is a multi-view database, consisting of 575 gray-scale images of 20 people (subject), each covering a wide range of poses from profile to frontal views. Each image has a resolution of 112×92 pixel. Each subject also covers a range of race, sex and appearance. Unlike the ORL database, the number of images per people is not fixed; it varies from 19 to 48. Fig. 4 shows some of the sample images of a subject from the database.

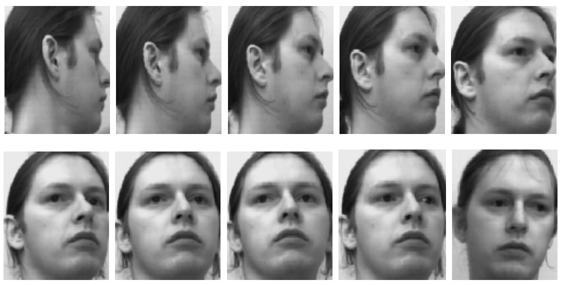


Fig. 4. Some sample images of a subject from the UMIST database

4.2.1 Randomly Partitioning the Database

Like AT&T database, we randomly select *d* images from each subject to form the training set and the remaining images are included in the test set. We choose the value of *d* as 4, 6, 8 and 10. It should be again noted that there is no overlap between the training and test images. For each value of *d*, experiment is repeated 20 times with

[1] At present UMIST database contains 475 images. However, we have used the earlier version of the UMIST database to test with more number of images.

different training and test sets. Fig. 5 shows the recognition rates of the G-2DFLD algorithm using a multi-class SVM. For each value *d*, average recognition rates are plotted by varying the values of *p* and *q*. For *d*=4, 6, 8 and 10 the best average recognition rates are found to be 86.22%, 92.28%, 95.54% and 96.92%, respectively and the dimension *(p×q)* of the corresponding image feature matrices are (14×14), (14×14), (14×14) and (18×18), respectively.

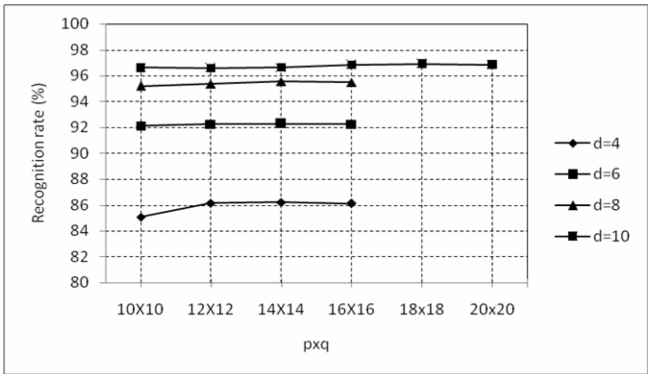


Fig. 5. Average recognition rates of the G-2DFLD algorithm on the UMIST database for different values *d* by varying the values of *p* and *q*

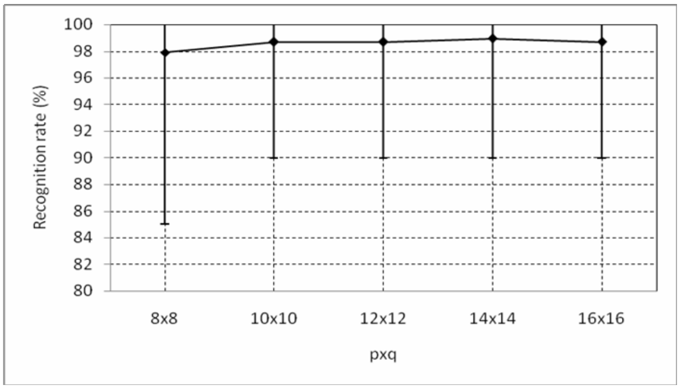


Fig. 6. Average recognition rates of the G-2DFLD algorithm on the UMIST database for 19-folds cross validation test by varying the values of *p* and *q*. The upper and lower extrema of the error bars represent the maximum and minimum values, respectively.

4.2.2 N-Fold Cross Validation Test

Since the number of images per subject varies from 19 to 48, we have randomly divided the database into 19 folds, taking one image of a subject into a fold. Therefore, in each fold there are 20 images, each one corresponding to a different subject. For 19-folds cross validation test, in each experimental run, 18 folds are used to train the multi-class SVM and remaining one fold is used for testing. Therefore, training and test sets consist of 360 and 20 images, respectively in a particular experimental run. The average recognition rates by varying the image feature matrix (i.e. $p\times q$) are shown in Fig. 6. The best average recognition rate is found to be 98.95% using image feature matrix of size (14×14).

4.2.3 Comparison with Other Methods

For a fair comparison, like AT&T database, we have implemented the PCA, 2DPCA, PCA+FLD and 2DFLD algorithms and used the same multi-class SVM and parameters for classification. The comparisons between the best average recognition rates of the PCA, 2DPCA, PCA+FLD and 2DFLD algorithms along with the propose G-2DFLD method using the two different experimental strategies are shown in Table 3. It may be again noted that in all the cases the performance of the G-2DFLD method is better than the PCA, 2DPCA, PCA+FLD and 2DFLD methods, excepts in 19-folds cross validation test, where the performance of the 2DPCA method matches with that of the proposed G-2DFLD method.

Table 3. Comparison of different methods in terms of average recognition rates (%) on the UMIST database. Figures within the parentheses denote the number of features.

Experiment	Method	*d*=4	*d*=6	*d*=8	*d*=10
Randomly partition, *d* images/subject	G-2DFLD	86.22 (14×14)	92.28 (14×14)	95.54 (14×14)	96.92 (18×18)
	PCA	80.72 (60)	86.53 (60)	94.01 (60)	95.11 (60)
	2DPCA	85.70 (112×14)	91.91 (112×14)	95.07 (112×14)	96.60 (112×18)
	PCA+FLD	76.31 (25)	85.69 (25)	90.93 (25)	93.72 (25)
	2DFLD	86.12 (112×14)	92.16 (112×14)	95.25 (112×14)	96.55 (112×18)
19-folds cross validation test	G-2DFLD	98.95 (14×14)			
	PCA	98.68 (60)			
	2DPCA	98.95 (112×14)			
	PCA+FLD	96.36 (25)			
	2DFLD	98.68 (112×14)			

5 Conclusion

In this paper, we have presented a novel scheme for face feature extraction, namely, generalized two-dimensional FLD (G-2DFLD) method, which is based on the original 2D image matrix. The G-2DFLD algorithm maximizes class separability from both the row and column directions simultaneously, resulting in a smaller image feature matrix. To realize this, we have defined two alternative Fisher's criteria corresponding to row and column-wise projection directions. The principal components extracted from an image matrix by the G-2DFLD method are scalars. Since the size of the scatter matrices in the proposed G-2DFLD algorithm is much smaller than those in the conventional PCA and FLD schemes, the computational time for feature extraction is much less. The experimental results on the AT&T and UMIST databases show that the G-2DFLD method is more efficient than the PCA, 2DPCA, PCA+FLD, and 2DFLD methods, not only in terms of computation times, but also for the task of face recognition using a multi-class support vector machine (SVM).

Acknowledgment. This work was partially supported by the UGC major research project (F. No.: 37-218/2009(SR), dated: 12-01-2010), CMATER and the SRUVM projects of the Department of Computer Science & Engineering, Jadavpur University, Kolkata, India. The author, Shiladitya Chowdhury would like to thank Techno India, Kolkata for providing computing facilities and allowing time for conducting research works. The author, D. K. Basu would also like to thank the AICTE, New Delhi for providing him the Emeritus Fellowship (F.No.: 1-51/RID/EF(13)/2007-08, dated 28-02-2008).

References

1. Fukunaga, K.: Introduction to Statistical Pattern Recognition. Academic Press, New York (1990)
2. Er, M.J., Wu, S., Lu, J., Toh, H.L.: Face recognition with radial basis function (RBF) neural networks. IEEE Trans. Neural Networks 13, 697–710 (2002)
3. Yang, J., Zhang, D., Frangi, A.F., Yang, J.-Y.: Two-dimensional PCA: A new approach to appearance-based face representation and recognition. IEEE Trans. Pattern Anal. Mach. Intell. 26, 131–137 (2004)
4. Belhumeur, P.N., Hespanha, J.P., Kriegman, D.J.: Eigenfaces versus fisherfaces: Recognition using class specific linear projection. IEEE Trans. Pattern Anal. Machine Intell. 19, 711–720 (1997)
5. Xiong, H., Swamy, M.N.S., Ahmad, M.O.: Two-dimensional FLD for face recognition. Pattern Recognition 38, 1121–1124 (2005)
6. ORL face database. AT&T Laboratories, Cambridge, U. K., `http://www.uk.research.att.com/facedatabase.html`
7. Graham, D.B., Allinson, N.M.: Characterizing Virtual Eigensignatures for General Purpose Face Recognition: From Theory to Applications. In: Wechsler, H., Phillips, P.J., Bruce, V., Fogelman-Soulie, F., Huang, T.S. (eds.) Computer and Systems Sciences. NATO ASI Series F, vol. 163, pp. 446–456 (1998)
8. Vapnik, V.N.: Statistical learning theory. John Wiley & Sons, New York (1998)

Learning Gradients with Gaussian Processes*

Xinwei Jiang[1], Junbin Gao[2,**], Tianjiang Wang[1], and Paul W. Kwan[3]

[1] Intelligent and Distributed Computing Lab,
School of Computer Science and Technology,
Huazhong University of Science and Technology, Wuhan, 430074, China
ysjxw@hotmail.com

[2] School of Computing and Mathematics,
Charles Sturt University, Bathurst, NSW 2795, Australia
jbgao@csu.edu.au

[3] School of Science and Technology,
University of New England, Armidale, NSW 2351, Australia
kwan@turing.une.edu.au

Abstract. The problems of variable selection and inference of statistical dependence have been addressed by modeling in the gradients learning framework based on the representer theorem. In this paper, we propose a new gradients learning algorithm in the Bayesian framework, called Gaussian Processes Gradient Learning (GPGL) model, which can achieve higher accuracy while returning the credible intervals of the estimated gradients that existing methods cannot provide. The simulation examples are used to verify the proposed algorithm, and its advantages can be seen from the experimental results.

1 Introduction

Analyzing data sets associated with many variables or coordinates has become increasingly challenging in many circumstances, especially in biological and physical sciences [1]. A wide range of machine learning algorithms based on the regularization theory such as support vector machines (SVMs) [2] have been proposed to solve the predictive problems in the past two decades. Although these approaches demonstrate quite acceptable and robust performances in a lot of experiments and applications, sometimes one also wants to get an insight into the relationships between the coordinates and the influence of the coordinates/attributes/features on the outputs. For example, it is very interesting to investigate which covariant is most significant for prediction and how the variables vary with respect to each other in estimation.

The gradient of the target function provides a valuable measure to characterize the relationships [3,4,5,1] and it has been used in many approaches and

* This research was supported by the National High Technology Research and Development (863) Program of China (2007AA01Z161).

** The author to whom all the correspondence should be addressed.

applications. For example, the minimum average variance estimation (MAVE) method and the outer product of gradients (OPG) estimation approach proposed by [3] focus on finding the effective dimension reduction (e.d.r.) space in the data sets by using the gradients of the training data implicitly and explicitly, respectively. These models show better performance in the estimation of e.d.r. space than others, but learning gradient information would fail in the "large m (dimension), small n (size of the dataset)" paradigm [6]. Recently, [4] and [5] proposed a method to learn the gradient of a target function directly from a given data set based on the Tikhonov regularization method, which avoided the overfitting problem in the "large m, small n" settings. The most significant statistical measure we can get by those nonparametric kernel based models is the gradient outer product (GOP) matrix, which can interpret the importance of the coordinates for the prediction and the covariation with respect to each other. In addition, with the assistance of spectral decomposition of the gradient outer product matrix, the e.d.r. directions can be directly estimated [1]. Furthermore [7] extended gradient learning algorithm from the Euclidean space to the manifolds setting, and provided the convergence rate dependent on the intrinsic dimension of the manifold rather than the dimension of the ambient space. This is very important in the "large m, small n" settings. Except for the application examples proposed in the available literature, gradient learning from scattered data sets is particularly important for surfaces reconstruction in computer graphics where, when visually scaled geometric surfaces constructed from scattered data, analytical expression (or rules) of gradients was highly desirable in calculating the normals at any given point needed in most surface reconstruction algorithms (see [8]).

However, these direct gradient learning methods cannot offer any reasonable error bars for the estimated gradients because essentially the task of estimating gradients is the problem of point estimation. In many application scenarios, a confidence interval on the estimates is very important, such as found in computer graphics.

In this paper, we propose a new gradient learning approach under the Bayesian framework based on Gaussian Processes (GPs) [9]. Compared to the learning gradients method in [4], not only can our algorithm apply in the "large m, small n" cases and achieve higher accuracy, but it can also return the error bars of the estimated gradients, which provide us with an estimate of the uncertainty of stability. We will verify these features in Sections 5.

The rest of this paper is organized as follows. In Section 2, we introduce the statistical foundation for learning gradients. The gradients learning method with Gaussian Processes will be proposed in Section 3, which includes a brief introduction of the Gaussian Processes regression. The algorithm derivation is illustrated in Section 4. Then, simulated data are used to verify our algorithm in Section 5. Finally, closing remarks and comments will be given in Section 6.

2 The Statistical Foundation for Learning Gradients

2.1 Notations

Denote data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{n}$ where $\boldsymbol{x}_i$ is a vector in a m-dimensional compact metric subspace $\mathcal{X} \subset \mathbb{R}^m$ and $\boldsymbol{y}_i \in \mathbb{R}^p$ is a vector too. Without loss of generality, we will assume that $p = 1$. Our approach can be easily extended to the case of vectorial value outputs $\boldsymbol{y}$. Typically we assume that the data are drawn i.i.d. from a joint distribution, $(\boldsymbol{x}_i, \boldsymbol{y}_i) \sim p(X, Y)$. In the standard regression problem, we want to model the regression function F defined by the conditional mean of $Y|X$, i.e., $F = \mathbb{E}_Y[Y|X]$. The gradient of F is a vectorial value function with m components, if all the partial derivatives exist,

$$\boldsymbol{f}(\boldsymbol{x}) \triangleq \nabla F = (f_1(\boldsymbol{x}), ..., f_m(\boldsymbol{x}))^T = \left(\frac{\partial F(\boldsymbol{x})}{\partial x^1}, \cdots, \frac{\partial F(\boldsymbol{x})}{\partial x^m} \right)^T \tag{1}$$

where x^i are the components of the vector $\boldsymbol{x}$ and $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), ..., f_m(\boldsymbol{x}))$.

The gradient and the issues of variable selection and coordinate covariation are relevant because the gradient can provide following information [4]:

1. Variable selection: the norm of the partial derivative $\left\| \frac{\partial F}{\partial x^i} \right\|$ indicates the significance of the variables for the prediction because a small norm implies a slight change in the function F with respect to the i-th coordinate.
2. Coordinate covariation: the inner product of the partial derivatives with respect to different dimensions $\langle \frac{\partial F}{\partial x^i}, \frac{\partial F}{\partial x^j} \rangle$ indicates the covariance of the i-th and j-th coordinates.

A central concept in all gradients learning approaches, called the gradient outer product (GOP) matrix, is defined by

$$\varGamma_{ij} = \mathbb{E}\langle \frac{\partial F}{\partial x^i}, \frac{\partial F}{\partial x^j} \rangle \tag{2}$$

The GOP has a deep relation with the so-called effective dimension reduction (e.d.r.) space and the relationship was exploited in several gradient regression methods such as MAVE and OPG [3] and [4,5].

2.2 Learning Gradients

To propose our approach for gradients learning, let us focus on the introduction of those available algorithms of learning the gradients from the data. Recall that the MAVE and OPG suffer from the problem of the overfitting in the "large m, small n" paradigm, so the so-called regularized framework has been used to overcome the overfitting based on the kernel representer theorem. Actually the kernel representer theorem is also the motivation for our algorithm in the next section. The algorithms based on the kernel representer theorem show better performance than the MAVE and OPG [1].

Our goal is to design a model for the gradient estimate directly from data. The conventional methods usually take a two-steps procedure by first learning a regression function F and then calculating the gradients of the F. However a direct gradients learning algorithm may have more advantages than the conventional ways, as demonstrated in [4,1].

Essentially, all of those kinds of models are motivated by the Taylor expansion of the target function:

$$y_i \approx y_j + \boldsymbol{f}(\boldsymbol{x}_i)^T(\boldsymbol{x}_i - \boldsymbol{x}_j) \quad \text{for} \quad \boldsymbol{x}_i \approx \boldsymbol{x}_j \tag{3}$$

A model for gradients leaning from an observation dataset $\mathcal{D}$ is defined as

$$\boldsymbol{f} := \operatorname*{argmin}_{\boldsymbol{f}} \left\{ \frac{1}{n^2} \sum_{i,j=1}^{n} w_{ij} [y_i - y_j + \boldsymbol{f}(\boldsymbol{x}_i)^T(\boldsymbol{x}_j - \boldsymbol{x}_i)]^2 + \lambda \|\boldsymbol{f}\|^2 \right\} \tag{4}$$

where w_{ij} is called weights and defined as $w_{ij} = \frac{1}{\sigma^{m+2}} \exp\left\{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right\}$ where σ^2 is set to the median of the input data. When $\boldsymbol{x}_j$ is far away from $\boldsymbol{x}_i$, the Taylor expansion of the function $F(\boldsymbol{x}_j)$ at $\boldsymbol{x}_i$ makes less contribution to the regression objective.

According to the representer theorem [10], the optimal solution to (4) is the linear combination of kernel function defined on the data points, thus the problem is actually transformed to solving a linear systems problem, see [4].

Due to regularization, this model can prevent overfitting in the "large m, small n" paradigm and obtain fairly remarkable performance. However, sometimes it is also important that we want to know the error bars of the point estimation for the gradient, which can not be provided by those kinds of models.

An alternative method is to define the model under the Bayesian learning and inference framework. We aim to use the Gaussian Processes (GPs) model which is also based on the kernel and can be viewed as the exhibition of the representer theorem. So motivated by the model in [4] and associated it with the GPs, we will show how to improve the accuracy and compute the error bars of the estimated gradient in the following section.

3 Gradients Learning with Gaussian Processes

3.1 Gaussian Processes Regression

Given the data set which consists of the i.i.d. samples from unknown distribution $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \cdots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\} \subset \mathbb{R}^m \times \mathbb{R}^p$ The standard Gaussian Process regression is concerned with the case when $p = 1$. The goal is to estimate the $p(\boldsymbol{y}|\boldsymbol{x}^*)$ for a test data $\boldsymbol{x}^*$. In the standard Gaussian processes (GPs) regression model, a latent variable f is introduced in the model defined by

$$y = f(x) + \epsilon$$

where ϵ is the additive noise, specified by the likelihood $p(y|f,x) = p(y|f)$. This model is nonparametric because the latent variable f is random function which follows the Gaussian Process with zero mean and covaiance function $k(\cdot,\cdot)$. Also the likelihood follows a Gaussian distribution with zero mean and covariance σ_t^2.

Denote by $X = \{\boldsymbol{x}_i\}_{i=1}^n$. Due to the independence of the samples $\mathcal{D}$, its likelihood under the model is the product of $p(y_i|f(\boldsymbol{x}_i))$ which is a Gaussian too. Given a test point $\boldsymbol{x}^*$, it is easy to check that the joint distribution of the latent function is given by, see [9],

$$\begin{bmatrix} f \\ f^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K_{XX} & K_{X\boldsymbol{x}^*} \\ K_{\boldsymbol{x}^*X} & K_{\boldsymbol{x}^*\boldsymbol{x}^*} \end{bmatrix}\right) \tag{5}$$

where K are matrix of the kernel function values at the corresponding points and $\mathcal{N}(\mu, \Sigma)$ denotes the Gaussian distribution with mean μ and covariance Σ.

Under the Gaussian likelihood assumption, we can simply add the covariance of the noise to the GP prior due to the independence assumption of the noise. So the predictive distribution on the observation is

$$f^*|\boldsymbol{x}^*, X, \boldsymbol{y} \sim \mathcal{N}(K_{\boldsymbol{x}^*X}(K_{XX} + \sigma_t^2\mathbf{I})^{-1}\boldsymbol{y},\ K_{\boldsymbol{x}^*\boldsymbol{x}^*} - K_{\boldsymbol{x}^*X}(K_{XX} + \sigma_t^2\mathbf{I})^{-1}K_{X\boldsymbol{x}^*}) \tag{6}$$

where the variance of the conditional distribution illustrates the uncertainty of the prediction and the mean can be written as $f(\boldsymbol{x}^*) = \sum_{i=1}^n \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}^*)$, where $\alpha = (K_{XX} + \sigma_t^2\mathbf{I})^{-1}\boldsymbol{y}$. This form of the prediction exhibits the fact that the GP can be represented in terms of a number of basis function is one feature of the representer theorem.

3.2 Gradients Estimation Model with Gaussian Processes

To apply the Gaussian Process model in the case of gradients learning, we have to overcome two hurdles. First, the regression model (4) shows that we are dealing with a multi-task regression problem as the gradient $\boldsymbol{f}$ is a vectorial function, so we have to generalize the standard Gaussian Process regression to multi-task case. This has been done in the recent works such as [11]. Second, the i.i.d. assumption for the data set can not be used to produce a joint likelihood which is the product of individual likelihood at each data point. In fact, when we transform (4) into probabilistic formulation, we see that the coupling between data makes learning and inference more complicated. However, we can still define a likelihood for the whole data set $\mathcal{D}$ rather than a likelihood for each data pair.

Under the above modification, we can formulate the gradients learning model in the Bayesian framework based on the GPs, named Gaussian Process Gradient Learning (GPGL) model, and we will show the advantages in Section 5.

Based on the analysis we have just given, a new likelihood formulation by extending the datum-based likelihood to dataset-based likelihood is defined as

$$p(Y|X, \mathbf{f}) \propto \exp\left\{ -\frac{1}{2}\sum_{i,j=1}^{n} w_{ij}[y_i - y_j + \boldsymbol{f}(\boldsymbol{x}_i)^T(\boldsymbol{x}_j - \boldsymbol{x}_i)]^2 \right\} \tag{7}$$

Let us introduce the following notation, the scalar $c = \sum_{i,j=1}^{n} w_{ij}(y_i - y_j)^2$, the $m \times m$ matrices $B_i = \sum_{j=1}^{n} w_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T$, and the m dimensional vectors $h_i = \sum_{j=1}^{n} w_{ij}(y_i - y_j)(\boldsymbol{x}_i - \boldsymbol{x}_j), i = 1, 2, ..., n$. We will use the same weights w_{ij} as [4] for comparison. Furthermore define $B = U^T \text{diag}(B_1, B_2, \cdots, B_n)U$ and a column vector of dimension mn $\boldsymbol{h} = U^T[h_1^T, h_2^T, \cdots, h_n^T]^T$, where U is a permutation matrix. Similarly define the column vector (of dimension mn) $\mathbf{f} = [\boldsymbol{f}_1^T, \boldsymbol{f}_2^T, ..., \boldsymbol{f}_m^T]^T$ where $\boldsymbol{f}_i = [f_i(x_1), f_i(x_2), ..., f_i(x_n)]^T$.

Under the above notation, it is easy to validate that the likelihood (7) of the observation dataset $\mathcal{D}$ can be written as

$$p(Y|X, \mathbf{f}) = \frac{1}{M} \exp\left\{ -\frac{1}{2}(\mathbf{f}^T B \mathbf{f} - 2\boldsymbol{h}^T \mathbf{f} + c) \right\} \tag{8}$$

where M is the normalized constant.

The variable $\mathbf{f}$ collects the information of m partial derivatives over the given input data X. In our model formulation, the variable $\mathbf{f}$ is assumed to be a Gaussian Processes while the covariance function is $\Sigma = K_{ff} \otimes K_{XX}$. So the Gaussian processes prior is

$$p(\mathbf{f}|X, \theta) = \frac{1}{|2\pi\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2}\mathbf{f}^T \Sigma^{-1} \mathbf{f} \right\} \tag{9}$$

where $K_{ff} \in \mathbb{R}^{m \times m}$ is the coordinate-similarity matrix, $K_{XX} \in \mathbb{R}^{n \times n}$ is the covariance matrix of the samples X, and θ is the parameters of the covariance function.

By using the Bayesian formulation, the posterior of $\mathbf{f}$ given the dataset is

$$p(\mathbf{f}|X, Y, \theta) = \frac{p(Y|X, \mathbf{f})p(\mathbf{f}|X, \theta)}{p(Y|X, \theta)} \tag{10}$$

As all the densities in the above relation are Gaussian, it is easy to derive, see Appendix A of [9], the posterior of $\mathbf{f}$

$$p(\mathbf{f}|X, Y, \theta) = \frac{1}{|2\pi E|^{1/2}} \exp\left\{ -\frac{1}{2}(\mathbf{f} - E\boldsymbol{h})^T E^{-1} (\mathbf{f} - E\boldsymbol{h}) \right\} \tag{11}$$

where $E = (B + \Sigma^{-1})^{-1}$.

For a new data $\boldsymbol{x}_*$, we want to estimate $\boldsymbol{f}_* = \boldsymbol{f}(\boldsymbol{x}_*)$ based on the observation data. According to the predictive distribution of Gaussian processes, we have

$$\boldsymbol{f}_*|\mathbf{f}, \boldsymbol{x}_*, X, \theta \sim \mathcal{N}((K^f \otimes K_{*X})^T \Sigma^{-1} \mathbf{f}, K^f \otimes K_{**} - (K^f \otimes K_{*X})^T \Sigma^{-1} (K^f \otimes K_{*X})) \tag{12}$$

where $K_{*X} = K(X, \boldsymbol{x}_*)$, $K_{**} = K(\boldsymbol{x}_*, \boldsymbol{x}_*)$. By integrating over the uncertainty $\boldsymbol{f}$ according to the posterior (11), we can get the gradients predictive distribution

$$\begin{aligned} p(\boldsymbol{f}_*|\boldsymbol{x}_*, X, Y) &= \int p(\boldsymbol{f}_*|\mathbf{f}, \boldsymbol{x}_*, X)p(\mathbf{f}|X,Y)d\mathbf{f} \\ &= \frac{1}{|2\pi Q|^{1/2}} \exp\left\{ -\frac{1}{2}(\boldsymbol{f}_* - P)^T Q^{-1}(\boldsymbol{f}_* - P) \right\} \end{aligned} \tag{13}$$

That is, the gradients predictive distribution is a Gaussian with the mean P and the covariance Q. Thus the gradient estimate is given by

$$P = (K_{ff} \otimes K_{*X})^T (B\Sigma + I)^{-1}\boldsymbol{h} \tag{14}$$

and the error bar is given by

$$Q = K_{ff} \otimes K_{**} - (K_{ff} \otimes K_{*X})^T (\Sigma + B^{-1})^{-1} (K_{ff} \otimes K_{*X}). \tag{15}$$

4 Learning Kernel Hyperparameters

To develop an approach for learning coordinate-similarity matrix K_{ff} and the kernel hyperparameters, we use gradient-based optimization of the marginal likelihood $p(Y|X,\theta)$. Without loss of generality, we just consider K_{ff} as unit matrix. Since K_{ff} controls the correlations between m dimensions of the gradients, the simplicity means that we are assuming the independence of different coordinates. Actually the optimization with respect to the parameters in K_{ff} can be dealt with in the same way as follows [11].

Then the log marginal likelihood $\log p(Y|X,\theta)$ is given by

$$L = -\frac{1}{2}\log|B^{-1} + \Sigma| - \frac{1}{2}\boldsymbol{h}^T(B^{-1} + \Sigma)^{-1}\boldsymbol{h} + C. \tag{16}$$

where C is a constant independent of the parameter θ, which can be ignored in optimizing L with respect to θ. To work out a formula for the derivatives of L with respect to θ, we refer to the matrix reference manual for the notation [12].

Denote by $F_1 = -\log|B^{-1} + \Sigma|$, then $dF_1 = -(B^{-1} + \Sigma)^{-1} :^T d(\Sigma) :$. Similarly, we have $dF_2 = \left((B^{-1} + \Sigma)^{-1}\boldsymbol{h}\boldsymbol{h}^T(B +^{-1} + \Sigma)^{-1}\right) :^T d(\Sigma) :$, where $F_2 = -\boldsymbol{h}^T(B^{-1} + \Sigma)^{-1}\boldsymbol{h}$,

According to the derivative formula for the Kronecker product of matrices, we have $d(\Sigma) = d(K_{ff} \otimes K_{XX}) = (I_{m,m} \otimes T_{n,m} \otimes I_{n,n})(K_{ff} : \otimes I_{n^2,n^2})dK_{XX} :$, where $T_{m,n}$, called the vectorized transpose matrix, is the $mn \times mn$ permutation matrix whose (i,j)-th elements is 1 if $j = 1 + m(i-1) - (mn-1)\lfloor \frac{i-1}{n} \rfloor$ or 0 otherwise.

So the derivatives of L with respect to θ is

$$\begin{aligned} \frac{\partial L}{\partial \theta} =& \frac{1}{2}\left[-((B^{-1} + \Sigma)^{-1} :)^T + ((B^{-1} + \Sigma)^{-1}\boldsymbol{h}\boldsymbol{h}^T(B^{-1} + \Sigma)^{-1} :)^T \right] \\ & (I_{m,m} \otimes T_{n,m} \otimes I_{n,n})(K_{ff} : \otimes I_{n^2,n^2})\frac{dK_{XX}}{d\theta}. \end{aligned} \tag{17}$$

In our experiments, we learn the parameters of the models so as to maximize the marginal likelihood using gradient-based search. The code is based on Neil D. Lawrence's MATLAB packages `Kern` and `Optimi`[1].

We have seen that $(B^{-1} + \Sigma)^{-1}$ needs to be inverted for both making predictions and learning the hyperparameters in time $\mathcal{O}(m^3n^3)$. This can lead to computational problems if mn is large. Although we only use cholesky decomposition and singular value decomposition to accelerate computation, the efficient approximation method in [11] can be directly used in our GPGL algorithm to reduce the computational complexity.

5 Experiments

In this section we will verify our GPGL algorithm in two simulated data sets to show the higher accuracy of the estimation and the credible intervals that the gradient learning methods in [4], named Mukherjee's algorithm in the following, can not gain. In the first data set, we generate some samples from four simple functions which can compute the real gradients for comparison. Another high-dimensional data set is used to test that our algorithm can be applied to show the variable selection and coordinate covariance like Mukherjee's algorithm.

5.1 Error Bar Estimation

We illustrate how GPGL can be used to estimate the credible intervals of the estimated gradient and compare Mukherjee's algorithm with GPGL to show higher accuracy that GPGL demonstrates.

Given four representative elementary regression models $y = \exp(x); y = \ln(x); y = x^2; y = \sin(x)$, where $\{(x_i, y_i)\}_{i=1}^n \in \mathbb{R} \times \mathbb{R}$ and $x_i \sim N(1, 0.1)$. In our experiment, we sampled 100 points from the Gaussian distribution. The true derivatives are given by $y' = \exp(x); y' = 1/x; y' = 2 * x; y' = \cos(x)$, respectively. The comparison of the results between proposed GPGL algorithm and Mukherjee's algorithm is shown in Figures 1 to 4. We use the mean squared error between the true derivative and learned derivative to measure the quality of learning algorithm. The smaller MSE means that a better performance of the algorithm. All the MSEs for those four functions with different algorithms are collected in Table 1. It can be seen that the proposed GPGL algorithm gives better performance in terms of lower MSEs for three out of the four functions.

Although for the functions $y = \exp(x)$ and $y = x^2$, Mukherjee's algorithm gives slightly better results, the proposed GPGL algorithm outperforms Mukherjee's Algorithm in other cases. However, in the experiment, we find that Mukherjee's algorithm is sensitive to the value of regularization parameter and the percentage of the eigenvalues parameters (see the code in [4]) that need to be chosen manually, especially the regularization parameter. Sometimes, it is hard to choose them optimally, although a standard cross validation can be

[1] `http://www.cs.manchester.ac.uk/~neil/software.html`

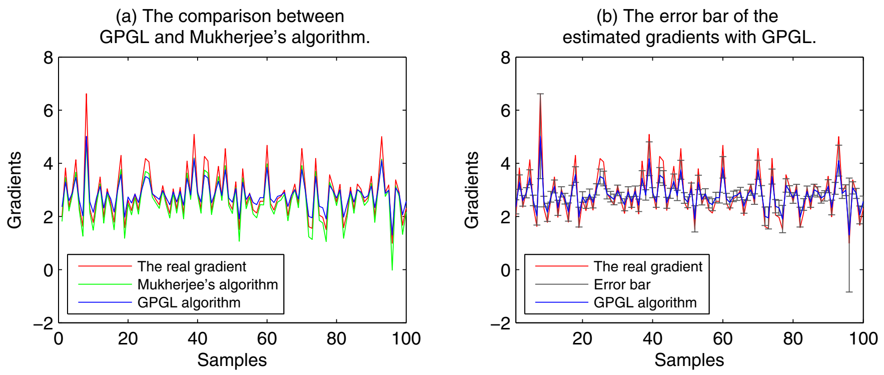


Fig. 1. The Result for function $y = \exp(x)$

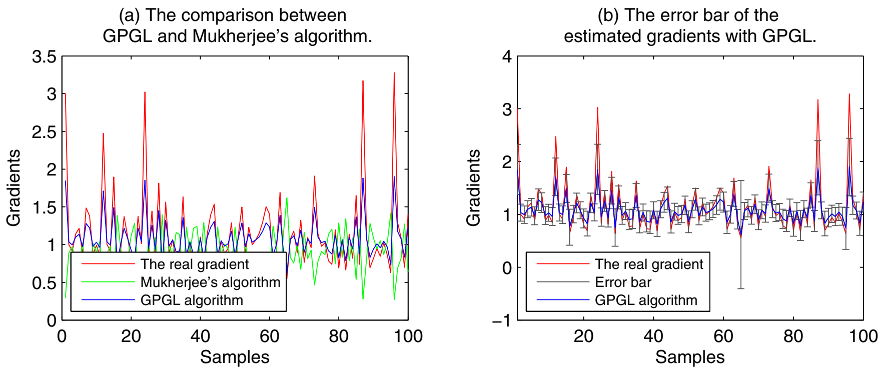


Fig. 2. The Result for function $y = \ln(x)$

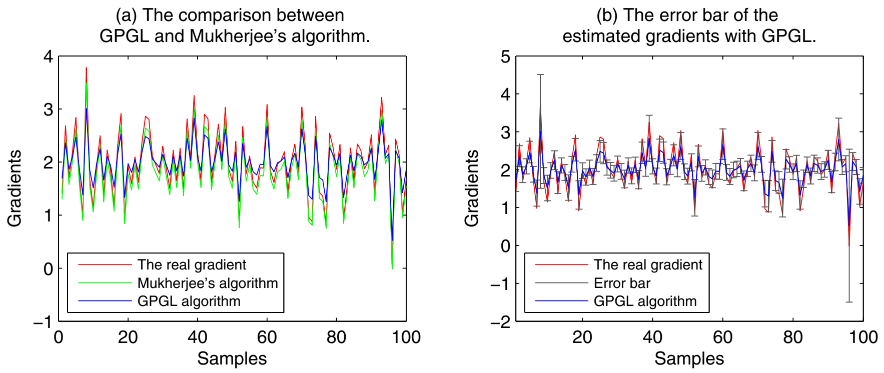


Fig. 3. The Result for function $y = x^2$

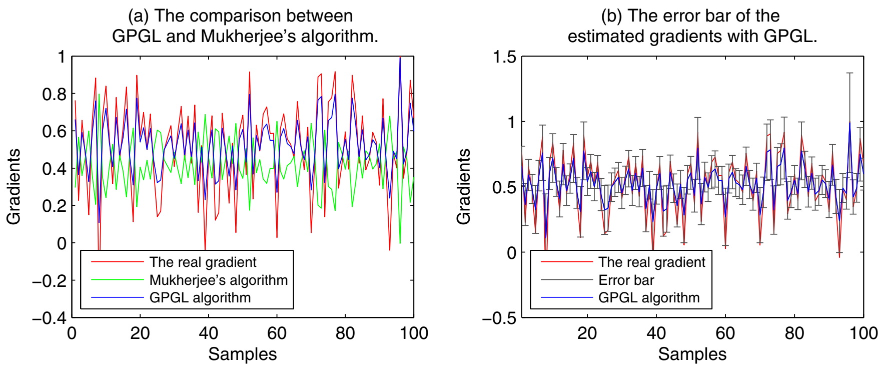


Fig. 4. The Result for function $y = \sin(x)$

Table 1. The Mean Squared Error

Algorithm	$y = \exp(x)$	$y = \ln(x)$	$y = x^2$	$y = \sin(x)$
GPGL	19.6275	12.8840	7.9557	1.3999
Mukherjee's	12.0330	80.8199	2.7621	15.9319

applied. However, the proposed GPGL method does not suffer from this problem and is more stable with ability to automatically adapt parameters. In addition, the error bars can be obtained from our algorithm along with gradient estimation.

5.2 High-Dimensional Data Set

Definition 1. The empirical gradient matrix (EGM), F_z, is the $m \times n$ matrix whose columns are $\boldsymbol{f}(\boldsymbol{x}_j)$ with $j = 1, \cdots, n$. The empirical covariance matrix (ECM), is the $m \times m$ matrix of inner products of the directional derivative of two coordinates, which can be denoted as $\mathrm{Cov}(\boldsymbol{f}) := [\langle (\boldsymbol{f})_p, (\boldsymbol{f})_q \rangle_K]_{p,q=1}^m$.

The ECM gives us the covariance between the coordinates while the EGM provides us information about how the variables differ over different sections of the space.

For a fair comparison, we construct the same artificial data as those used in [4]. By creating a function in an $m = 80$ dimensional space which consists of three linear functions over different partitions of the space, we generate $n = 30$ samples as follows:

1. For samples $\{\boldsymbol{x}_i\}_{i=1}^{10}$,

$$\boldsymbol{x}^j \sim \mathcal{N}(1, \sigma_x), \text{ for } j = 1, \cdots, 10;$$
$$\boldsymbol{x}^j \sim \mathcal{N}(0, \sigma_x), \text{ for } j = 11, \cdots, 80;$$

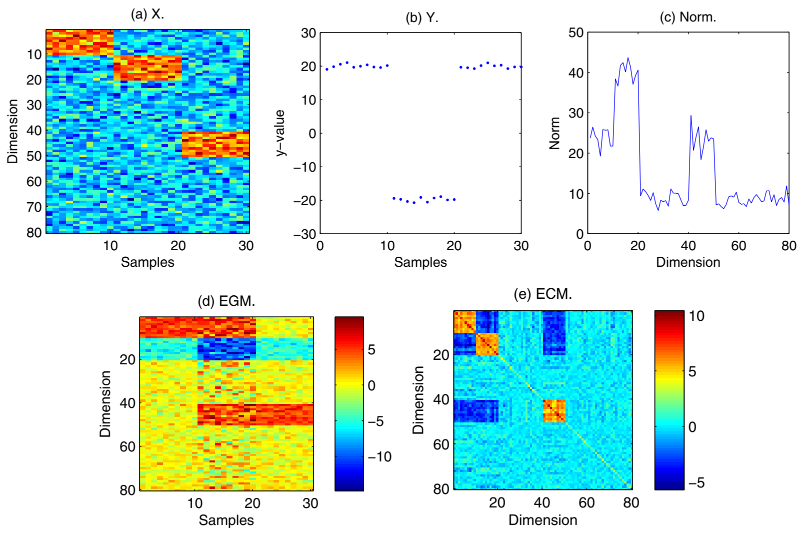


Fig. 5. a). The data matrix x; b). The vector of y values; c). The RKHS norm for each dimension; d). An estimate of the gradient at each sample; e). The empirical covariance matrix

2. For samples $\{\boldsymbol{x}_i\}_{i=11}^{20}$,

$$\boldsymbol{x}^j \sim \mathcal{N}(1, \sigma_x), \text{ for } j = 11, \cdots, 20;$$
$$\boldsymbol{x}^j \sim \mathcal{N}(0, \sigma_x), \text{ for } j = 1, \cdots, 10, 21, \cdots, 80;$$

3. For samples $\{\boldsymbol{x}_i\}_{i=21}^{30}$

$$\boldsymbol{x}^j \sim \mathcal{N}(1, \sigma_x), \text{ for } j = 41, \cdots, 50;$$
$$\boldsymbol{x}^j \sim \mathcal{N}(0, \sigma_x), \text{ for } j = 1, \cdots, 40, 51, \cdots, 80;$$

A representation of this X matrix is shown in Figure 5(a). Three vectors with support over different dimensions were constructed as follows:

$$w_1 = 2 + 0.5\sin(2\pi i/10) \text{ for } i = 1, \cdots, 10 \text{ and } 0 \text{ otherwise,}$$
$$w_2 = -2 + 0.5\sin(2\pi i/10) \text{ for } i = 11, \cdots, 20 \text{ and } 0 \text{ otherwise,}$$
$$w_3 = 2 - 0.5\sin(2\pi i/10) \text{ for } i = 41, \cdots, 50 \text{ and } 0 \text{ otherwise,}$$

Then the function is defined by

1. For samples $\{y_i\}_{i=1}^{10}$ $\quad y_i = \boldsymbol{x}_i \cdot w_1 + \mathcal{N}(0, \sigma_y)$,
2. For samples $\{y_i\}_{i=11}^{20}$ $\quad y_i = \boldsymbol{x}_i \cdot w_2 + \mathcal{N}(0, \sigma_y)$,
3. For samples $\{y_i\}_{i=21}^{30}$ $\quad y_i = \boldsymbol{x}_i \cdot w_3 + \mathcal{N}(0, \sigma_y)$.

A draw of the y values is shown in Figure 5(b). In Figure 5(c), we plot the norm of each component of the estimate of the gradient using the GPGL algorithm.

The norm of each component gives an indication of the importance of a variable and variables with small norms can be eliminated. Note that the coordinates with nonzero norm are the ones we expect, $l = 1, \cdots, 20, 41, \cdots, 50$. In Figure 5(d) we plot the EGM, while the ECM is displayed in Figure 5(e). The blocking structure of the ECM indicates the coordinates that covary. The similar result can be found in [4].

6 Conclusions

In this paper we have proposed a direct gradient learning algorithm from sample dataset in the Bayesian framework. The Gaussian Processes Gradient Learning (GPGL) model we propose can be seen as the manifestation of the representer theorem which is the basis of Mukherjee's algorithm. However, only the GPGL model can provide the error bars of the estimated gradients which characterize the uncertainty of the estimation. Besides, the GPGL model is stable and shows higher accuracy than Mukherjee's algorithm in terms of MSE in some circumstances. Another advantage is that GPGL model is more stable with automatical parameter adapting while the result from Mukherjee's algorithm heavily depends on the better tuning of the regularization parameters. In future work we plan to extend GPGL to sparse model to improve the generalization capability that is especially useful in the "large m, small n" setting.

References

1. Wu, Q., Guinney, J., Maggioni, M., Mukherjee, S.: Learning gradients: predictive models that infer geometry and dependence. Technical report, Duke University (2007)
2. Vapnik, V.: Statistical Learning Theory. Wiley, Chichester (1998)
3. Xia, Y., Tong, H., Li, W.K., Zhu, L.X.: An adaptive estimation of dimension reduction space. Journal of Royal Statistical Society 64(3), 363–410 (2002)
4. Mukherjee, S., Zhou, D.X.: Learning coordinate covariances via gradients. Journal of Machine Learning Research 7, 519–549 (2006)
5. Mukherjee, S., Wu, Q.: Estimation of gradients and coordinate covariation in classification. Journal of Machine Learning Research 7, 2481–2514 (2006)
6. West, M.: Bayesian factor regression models in the "large p, small n" paradigm. In: Bayesian Statistics, vol. 7, pp. 723–732. Oxford University Press, Oxford (2003)
7. Mukherjee, S., Wu, Q., Zhou, D.X.: Learning gradients and feature selection on manifolds. Technical report, Duke University (2007)
8. Dollar, P., Rabaud, V., Belongie, S.: Non-isometric manifold learning: Analysis and an algorithm. In: International Conference on Machine Learning, pp. 241–248 (2007)
9. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press, Cambridge (2006)
10. Schoelkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. The MIT Press, Cambridge (2001)
11. Bonilla, E.V., Chai, K.M.A., Williams, C.K.I.: Multi-task gaussian process prediction. In: Advances in Neural Information Processing Systems, vol. 20, pp. 153–160 (2008)
12. Brookes, M.: The Matrix Reference Manual (2005)

Analyzing the Role of Dimension Arrangement for Data Visualization in Radviz

Luigi Di Caro[1], Vanessa Frias-Martinez[2], and Enrique Frias-Martinez[2]

[1] Department of Computer Science, Universita' di Torino, Torino, Italy
dicaro@di.unito.it

[2] Data Mining and User Modeling Group, Telefonica Research, Madrid, Spain
{vanessa,efm}@tid.es

Abstract. The Radial Coordinate Visualization (Radviz) technique has been widely used to effectively evaluate the existence of patterns in highly dimensional data sets. A crucial aspect of this technique lies in the arrangement of the dimensions, which determines the quality of the posterior visualization. Dimension arrangement (DA) has been shown to be an NP-problem and different heuristics have been proposed to solve it using optimization techniques. However, very little work has focused on understanding the relation between the arrangement of the dimensions and the quality of the visualization. In this paper we first present two variations of the DA problem: (1) a Radviz independent approach and (2) a Radviz dependent approach. We then describe the use of the Davies-Bouldin index to automatically evaluate the quality of a visualization *i.e.,* its visual usefulness. Our empirical evaluation is extensive and uses both real and synthetic data sets in order to evaluate our proposed methods and to fully understand the impact that parameters such as number of samples, dimensions, or cluster separability have in the relation between the optimization algorithm and the visualization tool.

1 Introduction

Visualization tools focus on graphically representing high dimensional and multivariate data with enough clarity to allow for data exploration. Low dimensional data sets have traditionally been represented using either simple line graphs or scatter plots. Nevertheless, in the case of high dimensional data sets, special techniques for data visualization such as Parallel Coordinates [6], Star Glyphs [7], Circle Segments [2] or Radviz [11] are used. One of the key problems of these techniques is the dimension arrangement problem (DA), which evaluates from an algorithmic perspective which arrangement of the dimensions facilitates more the comprehension of the data. Ankerst *et. al* [1] formalized the DA problem and proved that it is NP-complete similarly to the traveling salesman problem. In this paper we present two reformalization of it designed to explore a search space whose non-convexity makes it more probable to find the desired global maxima (minima). The evaluation of the effectiveness of the arrangement in terms of visual information is typically carried out by means of human intervention. Most of the papers focusing on visualization techniques have generally assumed that the better the solution for the dimension arrangement optimization problem, the better the visual usefulness of the projected data. In this paper, we present an initial approach to formally determine such relation, making use of the Davies-Bouldin index for cluster analysis in order to compute the visual quality of the information being plotted by Radviz by an extensive empirical evaluation on synthetic and real datasets.

2 Related Work

There is a wide variety of visualization techniques for multidimensional data that present a circular arrangements of the dimensions, like Star Coordinates [7], Circle Segments [2] and Circle Graphs [14]. We focus our analysis on Radviz [4] which we further explain in Section 3. The problem of dimension arrangement is common for all circular and non-circular visualization techniques and was formalized by Ankerst *et al.* as an optimization problem where the similarity between dimensions located next to each other had to be maximized. to be NP-complete. So far, very little work has been done to automatically understand (without human intervention) the quality of the visualization for the projected data. Ankerst *et al.* evaluate the *goodness* of their dimension arrangement algorithms by simply stating that the results show clearly superiority. Yang *et al.* [12] proposed an interactive hierarchical ordering of the dimensions based on their similarities, thus improving the manageability of high-dimensional datasets and reducing the complexity of the ordering. Weng *et al.* [10] formalize the concept of clutter in various visualization techniques and present it as a dimension arrangement optimization problem whose solutions will improve the detection of structure in the projected data. Yang *et. al* [13] present a visualization technique where the user can interactively navigate through the dimensions and visually determine the quality of the re-arrangement. VizRank [9] is one of the few works that attempts to automate the visual quality evaluation process, by assessing data projections and ranking them according to their ability to visually discriminate between classes. The quality of the class separation is estimated by computing the predictive accuracy of the k-nearest neighbour classifier. Our evaluation scheme is faster and simpler than the VizRank approach and does not suffer from the typical k-NN problems such as the computation of an adequate value for k or the computational complexity ($O(n^2)$).

3 Radviz's Algorithm

RadViz (Radial Coordinate visualization) [4][5] is a visualization technique based on Hooke's law that maps a set of n-dimensional points into a plane: each point is held in place with springs that are attached at the other end to the feature anchors. The stiffness of each spring is proportional to the value of the corresponding feature and the point ends up at the position where the spring forces are in equilibrium. Prior to visualization, feature values are scaled to lie between

0 and 1. Radviz offers a unique method which can help to identify relations among data. Its main advantage is that it needs no feature projections and provides a global view on the multidimensional, multivariate data. The condition of equilibrium for a single object $\boldsymbol{u}$ is given by $\sum_{i=0}^{n-1}(\boldsymbol{A}_i - \boldsymbol{u}) * y_i = 0$.

Radviz faces several open problems: *overlapping* (different objects can be placed in the same 2D point), *visual clutter* (different instances could be placed close to each other) and *NP-completeness* (the final effectiveness of the approach depends on the dimension arrangement). Despite that, no study has shown yet whether there exists a relation between the solution provided by the optimization algorithm and the improvement in the visual usefulness of the projection.

4 Dimension Arrangement Formalizations

Although the DA problem has already been formalized in a generic context by Ankerst *et al.* [1], here we present new formalizations within the context of Radviz with the goal of providing a better exploration of the search spaces.

4.1 Independent DA

Let us assume that we have a dataset with points m that represent information represented with d dimensions. We define the similarity matrix as a symmetric matrix of dimensions $d \times d$, where each element $S_{i,j}$ represents the similarity between dimensions i and j. Each dimension i is represented as a distribution of m elements, where each element is taken from the $i-th$ dimension of each point in the dataset. In the experimental section we will describe the various metrics we have used to compute such similarity metric. Additionally, we define the neighborhood matrix N of dimensions $d \times d$which describes the circular distance between any two dimensions located in the circle. In particular, we calculate each $N_{i,j}$ as $1 - \frac{cdist(i,j)}{(d/2)}$, where d is the total number of dimensions and $cdist(i,j)$ represents the circular distance between dimensions i and j located on the circle. This distance is calculated as the number of dimensions on the circle between i and j through the shortest circular path. The larger the value of $N_{i,j}$, the closer the dimensions i and j are on the circle.

Thus, we can then formalize the dimension arrangement problem for a pair of similarity matrix S and neighborhood matrix N as a maximization problem where $\sum_{i=0}^{d-1}\sum_{j=0}^{d-1} N_{i,j} * S_{i,j}$ achieves its maximum value (*i.e.,* the more similar two dimensions are, the closer they should be located in the arrangement.

4.2 Radviz-Dependent DA

Our second DA formalization focuses on using Radviz to evaluate the quality of the arrangement. Again, we start with the similarity matrix S of dimensions $d \times d$. For each possible dimension arrangement, this matrix represents a measure of the similarities across dimensions. For each specific matrix S, we project each row in S onto the circle using Radviz. The idea is that the projected dimension should be as close to its dimension on the circle as possible. If that does not happen, it may be either that the dimensions are highly correlated or that the dimension arrangement is not good. Thus, for each dimension arrangement, each dimension i in the graph will have two representations: its coordinates on the circle, and its projected coordinates inside the circle, where the arranged dimensions are located according to the angular positions and the projected dimensions are calculated with respect to the Radviz formula.

Thus, the dimension arrangement problem can be defined as an optimization problem where for a given similarity matrix S, the optimal dimension arrangement is given by minimizing the sum of Euclidean distances between the arranged and the projected dimensions within the graph. This formalization follows the fact that the shorter the distance between an arranged dimension and its projection, the better the quality of the arrangement.

5 Experimental Setting

We want to focus our analysis on the relationship between the multiple DAs, the optimization functions, and the quality of visualization. For that purpose, in our analysis we make use of datasets with a limited number of dimensions that will allow us to fully explore, through a brute-force analysis, all the range of possible solutions. Our aim is twofold:

– To understand whether the formalization of the optimization problem as well as the metrics to measure similarity play a role in the way the search space (of the dimension arrangements) is explored.
– To carry an extensive experimental evaluation with both real and synthetic datasets to determine the relationship between the dimension arrangements and the quality of their associated data projections, studying the impact of various parameters like number of instance, dimensions, classes, and overlapping of the classes.

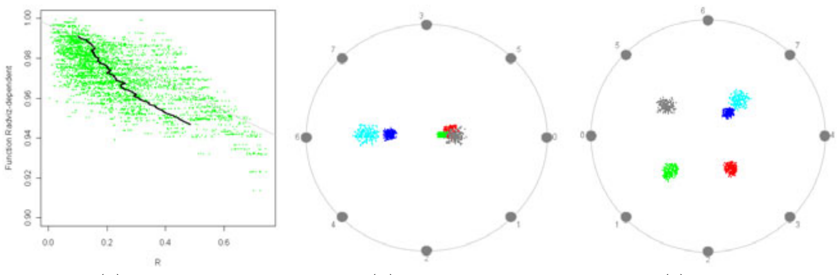
Regarding the synthetic data generation, we define four parameters for our algorithm: the number of classes nc (values from 2 to 100); the number of dimensions nd[1]; the number of instances ni (values from 100 to 10000) and the percentage $p\_overlap$ (up to 40%)[2] of instances that are randomly moved from one class to another. For each possible combination of nc, nd and ni, we create random instances within each class such that the clusters representing the classes are initially separated by equal distances. Finally, we modify the membership of a percentage $p\_overlap$ of the instances such that the boundaries between classes become blurry and classes start to overlap. We then used several real datasets from the UCI Machine Learning Repository[3].

The DA formalizations we have proposed are based on similarity measurements between dimensions. Although there exist many metrics to measure the

[1] We imposed a max # of dimensions (8) to be able to fully explore all possible DAs.
[2] Larger values did not add any extra overlap and were not considered.
[3] Datasets available at `http://archive.ics.uci.edu/ml/datasets.html`

similarity, we make use of the the Kullback-Leibler divergence [15] and the Cosine Similarity. The Kullback-Leibler (KL) divergence measures the difference between two probability distributions P and Q with cardinality d[4]: $\sum_{i=1}^{d} P_i * log_2(\frac{P_i}{Q_i})$. The inverse of it represents the similarity between them. On the other hand, Cosine similarity is calculated as the dot product between the distributions P and Q divided by the product of their norms. In order to study the relationship between a dimension arrangement and the visual usefulness of its projected data in Radviz, we first need to determine how visual usefulness is measured. The quality of the projected data onto the circle is related to the quality of the clusters obtained *i.e.,* the better the separation across clusters and instances, the more information the visual representation will convey to the data analyst. Thus, we measure visual usefulness of a data projection (and its corresponding dimension arrangement) using the Davies-Bouldin index (DB) [3]. DB is known to be one of the best indices to measure both the inter- and intra-cluster separation [8]. The DB index is computed as $\frac{1}{n} * \sum_{i=1}^{n} \overset{i \neq j}{max} \left\{ \frac{S_n(Q_i)+S_n(Q_j)}{S_n(Q_i,Q_j)} \right\}$, where n is the number of clusters, $S_n(Q_i)$ is the average Euclidean distance from each instance in cluster Q_i to its cluster centroid, and $S(Q_i, Q_j)$ is the Euclidean distance between cluster centers (Q_j can be any one of the clusters, a part from Q_i). Hence, the smaller the ratio, the more compact and separated the clusters are. Consequently, we seek dimension arrangements whose corresponding data projections have small DB indices associated. However, it may be the case that an initial dataset of instances with d dimensions shows a very high DB index in the d dimensional space, and thus it becomes very hard for its projected dataset to offer a good visualization. Thus, instead of measuring the DB of a projection, we measure the ratio R between the DB in the original data and the DB in the 2-dimensional mapping. Higher values of R correspond to higher visualization quality of the projected data. The first objective of the experimental evaluation is to be able to determine the relationship between the dimension arrangement and the quality of the associated visualization for each combination of the following parameters: (i) a specific dataset, either real or synthetic, (ii) a specific formalization of the DA problem, and (iii) a specific metric. Figure 1(a) shows an example result with the function *Radviz-dependent.* The number of points represents the number of dimension arrangements, while the black line represents the average values of a sliding window that captures the trend of density. We can observe that low values of the optimization function correspond to high values of R (minimization problem). The relation will be inverse when considering the function *independent* (maximization problem). Figures 1(b) and 1(c) show the Radviz projections associated to the worst value of R and the best value of R respectively. In Figure 2(a) we can observe that as the number of samples in the initial dataset increases, the best visual quality values R for the projected data decreases logarithmically. Figure 2(b) shows the visual information R versus the value of the optimization function for all possible DAs of datasets with 4, 6, and 8 dimensions. We can infer a general trend whereby as the number

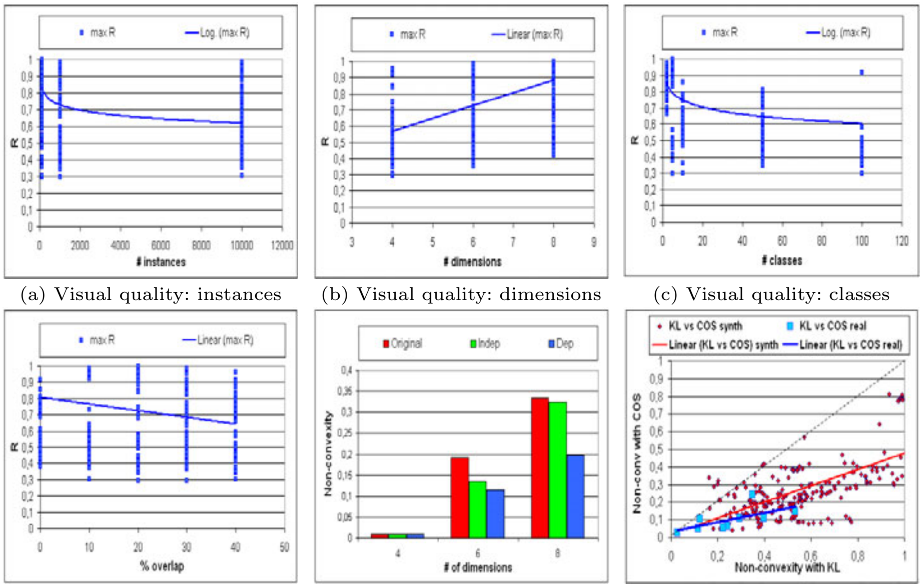
[4] We used a symmetric version of the original Kullback-Leibler divergence.



(a) Trend (b) Worst R (c) Best R

Fig. 1. (a) shows the correlation between the optimization function *Radviz-dependent* and the visual usefulness R of the DAs (the green points), using a synthetic dataset with 5 classes, 1000 instances, 8 dimensions and 10% of overlap. (b) and (c) show the projections with the best and the worst value of R.

of dimensions increases, the visual usefulness also improves following a linear curve. This result implies that as the number of dimensions grow, the Radviz technique manages to better maintain the initial distribution of the dataset *i.e,* the more dimensions, the better the samples can be characterized and the better Radviz will perform. Furthermore, this result confirms previous reports stating that the Radviz technique is useful for highly dimensional datasets [11]. Figure 2(c) shows the visual quality R of the Radviz projections of datasets containing from 5 to 100 different classes. Similarly to the number of instances, we observe that as the number of classes increases, the quality of the projected data decreases logarithmically. Figure 2(d) that the maximum value of R is linearly reduced as the percentage of overlap increases (a bad Radviz projection may be bad because of the technique itself or may be bad due to the fact that the initial dataset is hardly separable). Thus, the computation of the DB index for the initial dataset can give us an insight on how well the Radviz visualization can do. Moreover, we want to understand whether the formalization of the optimization function that explores the DA has an impact in the way the optimal solution is obtained. The optimization function associates a numerical value to each of the DAs. Our Independent function (*indep*) looks for the highest value (maximization problem), and our dependent function (*dep*) looks for the smallest value (minimization problem). In order to understand the *quality* of the search space, we evaluate its non-convexity. The non-convexity of the search space gives a measure of the probability that the optimization function will fall into a local minima. The smaller the non-convexity of the search space, the higher the probability of a local minima (or maxima) being a global minima (or maxima). We calculate the non-convexity of the search spaces using the Haussdorf distance as $\lambda(A) = sup_{x \in co\,A}\ inf_{y \in A} \parallel x - y \parallel$ where A is the set of points in the space search and $co\ A$ represents its convex hull. We compute the non-convexity for the DA formalizations presented in Section 4: Independent function *indep*, Radviz-dependent function *dep* and the binary neighborhood matrix initially described by Ankerst *et al.* [1] (referred from now on as *Original*). As we can observe in Figure 2(e), the *Original* function presented in [1] has a search area that is much less convex than the other two optimization functions for all possible combination of parameters and datasets. Still, such gap grows as the number of dimensions increases. These results indicate a higher probability of finding a global minimum (or maximum) when using the formalizations proposed in this paper.From previous analysis, the metric does not seem to impact the visual quality of the projections in terms of number of instances, classes, dimensions or percentage of overlap. In fact, we observe similar R values for both KL and COS across all the analysis. However, we want to understand whether the metric has an impact in the way the search space is explored *i.e.,* whether the selection of a metric can help decrease the chances of the optimization algorithm falling into a local minima (or maxima). For that purpose, we compute the non-convexity of the search spaces explored when using any combination of parameters. Figure 2(f) shows the trend between the non-convexity values of all combinations of parameters for each KL and COS metric. We can observe that the COS metric has smaller non-convexity values than KL. Hence, although in principle both metrics can potentially find solutions with similar visual quality R, the COS metric decreases the chances of the optimization function falling into local minima (or maxima), thus increasing the probability of finding a better DA.



(a) Visual quality: instances (b) Visual quality: dimensions (c) Visual quality: classes

(d) Visual quality: overlap (e) Non-convexity (opt.functions) (f) Non-convexity (metrics)

Fig. 2. Impact of the parameters in the visual quality of the projections: (a) instances, (b) dimensions, (c) classes, and (d) overlap; analysis of non-convexity according to (e) optimization functions and (f) metrics with both real and synthetic data

6 Conclusions

Radviz (and radial visualizations) is one of the most common techniques to help in the process of detecting patterns when visualizing high dimensional data. One of the main problems of these techniques is that the usefulness of the projections highly depends on the dimension arrangement (DA), which is a NP-complete problem. In this paper, we have presented two novel variants for the formalization of the DA problem showing that they allow to explore a search space whose non-convexity makes it more probable to find the desired global maxima (minima). Then, we have presented a technique to automatically evaluate the visual usefulness of a projection by means of the Davies-Bouldin index, studying the relationships and the impact of various metrics and parameters in the quality of the visualization.

References

1. Ankerst, M., Berchtold, S., Keim, D.A.: Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In: INFOVIS (1998)
2. Ankerst, M., Keim, D.A., Kriegel, H.-P.: Circle segments: A technique for visually exploring large multidimensional data sets. In: Visualization (1996)
3. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI 1(2), 224–227 (1979)
4. Hoffman, P., Grinstein, G., Marx, K., Grosse, I., Stanley, E.: Dna visual and analytic data mining. In: VIS (1997)
5. Hoffman, P., Grinstein, G., Pinkney, D.: Dimensional anchors: a graphic primitive for multidimensional multivariate information visualizations. In: NPIVM (1999)
6. Inselberg, A., Dimsdale, B.: Parallel coordinates: a tool for visualizing multidimensional geometry. In: VIS, Los Alamitos, CA, USA, pp. 361–378 (1990)
7. Kandogan, E.: Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In: IEEE Information Visualization Symp. (2000)
8. Kovács, F., Iváncsy, R.: Cluster validity measurement for arbitrary shaped clusters. In: AIKED, Wisconsin, USA (2006)
9. Leban, G., Zupan, B., Vidmar, G., Bratko, I.: Vizrank: Data visualization guided by machine learning. Data Min. Knowl. Discov. 13(2), 119–136 (2006)
10. Peng, W., Ward, M.O., Rundensteiner, E.A.: Clutter reduction in multi-dimensional data visualization using dimension reordering. InfoVis (2004)
11. Sharko, J., Grinstein, G., Marx, K.A.: Vectorized radviz and its application to multiple cluster datasets. In: Visualization and Computer Graphics. IEEE, Los Alamitos (2008)
12. Yang, J., Peng, W., Ward, M.O., Rundensteiner, E.A.: Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. In: Proc. IEEE Symposium on Information Visualization (2003)
13. Yang, J., Ward, M.O., Rundensteiner, E.: Visual hierarchical dimension reduction for exploration of high dimensional datasets (2003)
14. Aumann, Y., Feldman, R., Yehuda, Y.B., Landau, D., Liphstat, O., Schler, Y.: Circle graphs: New visualization tools for text-mining. In: Żytkow, J.M., Rauch, J. (eds.) PKDD 1999. LNCS (LNAI), vol. 1704, pp. 277–282. Springer, Heidelberg (1999)
15. Zhu, H.: On information and sufficiency (1997)

Subgraph Mining on Directed and Weighted Graphs

Stephan Günnemann and Thomas Seidl

Data management and data exploration group
RWTH Aachen University, Germany
{guennemann,seidl}@cs.rwth-aachen.de

Abstract. Subgraph mining algorithms aim at the detection of dense clusters in a graph. In recent years many graph clustering methods have been presented. Most of the algorithms focus on undirected or unweighted graphs. In this work, we propose a novel model to determine the interesting subgraphs also for directed and weighted graphs. We use the method of density computation based on influence functions to identify dense regions in the graph. We present different types of interesting subgraphs. In experiments we show the high clustering quality of our GDens algorithm. GDens outperforms competing approaches in terms of quality and runtime.

1 Introduction

Today's complex data can often be described by graphs. The nodes in a graph are objects while the edges illustrate connections between the objects. Examples include biological networks or social networks. A common property of such graphs is the existence of densely connected subgraphs. These clusters or communities are separated by less dense regions in the graph. We can gain a benefit of finding such interesting subgraphs. Based on a biological network, the development of useful drugs deduced from functional modules in protein interaction networks, is one example. In online commercial systems one can use interesting subgraphs for target delivery of customers. Customers in the same interesting subgraph show similar behavior.

Beside the connections between the objects in many cases also the directions and the weights are given. Let us consider a graph that represents the network traffic e.g. of the Internet. Edges that connect the routers of an ISP usually have higher weights/traffic amount than edges to nodes reflecting an end-user PC. These weights are important to identify the core and hence the dense subgraph of the total graph. Second, in general end-users generate more downlink traffic than uplink traffic; thus, the ingoing and outgoing edges are not identical and should be treated separately. Another example is an author graph where the edge weights can be interpreted as the number of co-written papers and the direction as the first author vs. co-author relationship. Overall, the identification of interesting subgraphs based on directed and weighted graphs is an important research field.

Related Work. Several graph mining algorithms have been presented to the community. The identification of optimal dense subgraphs based on some objective function is usually a hard problem [1], so that approximations or simple models are used. Some simple models for the detection of dense subgraphs include the identification of cliques or more meaningful of quasi-cliques [2, 3]. These algorithms usually generate a huge output, even if we retain only the maximal quasi-cliques, and the subgraphs overlap to a very high extend, resulting in marginal differences between subgraphs. Furthermore only undirected and unweighted graphs are used.

Another area is graph partitioning. Algorithms from this area try to divide the graph in flat parts; within these parts the nodes are strongly connected while between different subgraphs only loose connectivity exists. Models based on the maximum flow principle [4] or the k-partite graph partitioning [5] follow this paradigm. Another approach using physical principles is presented in [6]. Further techniques are the spectral clustering [7] or relational clustering [8]. One problem is that usually the number of interesting subgraphs or the size of the groups must be given. Furthermore, each node belongs to or is considered as a cluster even it is not well suited for this. Thus, the SCAN model [9] additionally identifies hubs or outliers; noise nodes are not included in the clusters. Directed and weighted graphs are not considered.

Moreover, several hierarchical/recursive methods exist that split up the graph step by step in smaller subgraphs, e.g. based on the cut-principle [10–12]. Paradigms based on edge betweenness are also used [13] and were extended to allow overlapping subgraphs [14, 15]. However, expensive recalculations are often performed. The modularity [16–18] is another well know measure, which is used for the recursive identification of subgraphs. All methods generate a complete hierarchy of interesting subgraphs. Each cut through this hierarchy represents a more or less meaningful partitioning of the graph in interesting subgraphs. However, many cuts are possible and the user is cluttered with a huge amount of subgraphs. Additionally, the construction of a complete hierarchy results in a high runtime.

Our Contributions. Our model is able to determine the interesting subgraphs on directed and weighted graphs. The direction of edges is a particular aspect, which leads to different definitions of dense subgraphs. Furthermore we present a definition of our interesting subgraphs based on the principle of density calculation. The number of clusters and their sizes are automatically determined and the clusters are not obfuscated by noisy nodes, i.e. nodes that do not belong to any interesting subgraph.

2 Mining Interesting Subgraphs

In this section we present our model for the mining of interesting subgraphs. Section 2.1 starts with some preliminaries. In Section 2.2 the density calculation on graphs is described, followed by our dense subgraph definitions in Section 2.3. In Section 2.4 algorithmic aspects are discussed and Section 2.5 concludes with some complexity results.

2.1 Preliminaries

A directed and weighted graph G is a tuple (V, E, w) with nodes V, edges $E \subseteq V \times V$, and a weighting function $w : E \rightarrow \mathbb{R}^+$. A path in G is a list of nodes $< p_1, \ldots, p_n >$ with $(p_i, p_{i+1}) \in E$. A node v is reachable from u along a set of nodes M, if there exists a path $< p_1, \ldots, p_n >$ with $p_1 = u$, $p_n = v$ and $p_i \in M$. Usually $M = V$, i.e. one can use all nodes in the graph for the path.

For density calculation, each object influences any other object with a certain value. The sum of influences determines the density of an object/a point in the data space. The influence value is usually based on two criteria. First, one calculates the distance between the objects, e.g. for two vectors o, p the Euclidean distance $d_2(o, p)$. Second, the distances are weighted according to a weighting function $\mathcal{W} : \mathbb{R} \rightarrow \mathbb{R}$. For this purpose often kernel functions are used as the Uniform, the Gaussian or the Epanechnikov kernel. The Epanechnikov kernel is known to be efficient and effective [19]. The overall influence of an object o on another object p is obtained by $influence(o, p) = \mathcal{W}(\frac{d(o,p)}{h})$. The factor h is used to scale the distances. The smaller the distance between two objects the higher is their influence on each other. The overall density of an object p is then calculated as the sum of influences $influence(o, p)$ for each object o in the database.

2.2 Density Computation on Graphs

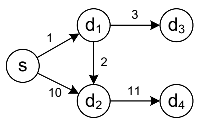
The challenge in our task is to consider the underlying graph structure to define a meaningful density calculation. Thus, in a first step we have to use graph based distances. Nodes that are 'stronger' connected, i.e. are more similar with respect to the selected distance function, should have a higher influence on each other. For ease of presentation we assume that smaller edge weights correspond to higher connectivity of the nodes. If the reverse is true, e.g. the number of co-written papers should be high for a strong connection between two nodes in an author graph, we can simply transform the weights to $1/w(u, v)$. In our approach we use the shortest path distance between two nodes s, d as a representative for a graph based distance function.

Definition 1. *Shortest path distance and influence*
Given a weighted graph $G = (V, E, w)$, the shortest path distance between node s and d is defined as

$$d_{min-path}(s, d) = \min_{path\ <p_1,\ldots,p_n>} \{\sum_{i=1}^{n-1} w(p_i, p_{i+1}) \mid p_1 = s \wedge p_n = d\}$$

The influence based on the Epanechnikov kernel is defined by

$$influence(s, d) = \mathcal{W}(\tfrac{d_{min-path}(s,d)}{h}) \text{ with } \mathcal{W}(x) = \begin{cases} \frac{3}{4}(1 - x^2) & |x| \leq 1 \\ 0 & else \end{cases}$$



$$d_{min-path}(s,d_1)=1$$
$$d_{min-path}(s,d_2)=3$$
$$d_{min-path}(s,d_3)=4$$
$$d_{min-path}(s,d_4)=14$$

$$influence(x,y)>0 \text{ iff } d_{min-path}(x,y)<5$$

Fig. 1. Influence region and direct influence

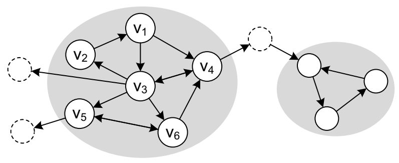


Fig. 2. Cluster cores

By this definition the influence decreases quadratically with the distance and the influence of a node s is not restricted to its direct neighbors but we affect also nodes that are reachable over longer paths. Thus, the connectivity of the nodes is more precisely reflected. In Figure 1 the node d_3 is influenced by s even though it is not a direct neighbor. Furthermore, we do not simply count the number of near located nodes but we weight them according to their distance. Or even stronger, with the non-linear weighting of $\mathcal{W}$ a single very close node results in a higher influence than two not as close nodes. Due to the compact support of the Epanechnikov kernel $\mathcal{W}$ we only have to calculate the shortest path distances up to h, i.e. we do not need to analyze all nodes; in contrast to functions with a non-compact support as the Gaussian kernel. Thereby we increase the efficiency of our method. In contrast to classical density computation our influence function need not to be symmetric, i.e. $influence(s,d) \neq influence(d,s)$ is possible. This non-symmetry is particularly appropriate for directed graphs.

Another important aspect in graphs is the possible influence on nodes that are at a first glance not influenced. In Figure 1 for example the edge between the nodes s and d_2 has a too high weight; thus, s does not influence d_2 based on this edge. However, we can use a 'detour' including other nodes to get an positive influence. Therefore, we distinguish two different node sets: First, the overall set of nodes on which the node s has influence on, the so called influence region. Second, the directly influenced nodes that correspond to closely located neighboring nodes.

Definition 2. *Influence region and direct influence*
Given a graph G and a node s, the influence region $region(s)$ of s is defined by

$$d \in region(s) \Leftrightarrow influence(s,d)>0$$

The set $direct(s)$ contains all nodes that are directly influenced by s and is defined by

$$d \in direct(s) \Leftrightarrow (s,d) \in E \wedge w(s,d)<h$$

This distinction is based on the intuition that one only interacts with good friends that are directly known to oneself, e.g. to spread a rumor. However, a rumor spreaded by a person can also reach persons not only in its neighborhood, but along a path of good friends. In Figure 1 we get $region(s)=\{s,d_1,d_2,d_3\}$ and $direct(s)=\{d_1\}$ and obviously $direct(s) \subseteq region(s)$ holds. By our definition, the influence region is connected. Even stronger, s can reach all other nodes in its influence region.

Now we are able to calculate the density of a node d. We have to determine all nodes that have an influence on d; in contrast to the nodes which d itself influences, according to the non-symmetry. We call this set of objects the reverse influence region and define $revRegion(d) = \{s \in V \mid d \in region(s)\}$. The density of a node d is the sum of all influences from the objects in this set.

Definition 3. *Density of a node*
The density of a node d is defined by

$$density(d) = \sum\nolimits_{s \in revRegion(d)} influence(s, d)$$

In Figure 1 the reverse influence region of node d_2 is $\{s, d_1, d_2\}$. The actual density can be calculated based on the individual influence values. The higher the density the more interesting is the node in our graph.

2.3 Density Based Clusters on Graphs

Our idea is to identify interesting subgraphs, i.e. subgraph clusters, by dense areas that are separated by sparse areas. For vector spaces similar approaches show good performance even in the presence of noise [20, 21]. We will call a node dense if its density exceeds a threshold τ. All nodes fulfilling this minimal density criterion are the core nodes, as these nodes are the candidates that build the cores of different clusters.

Definition 4. *Core nodes*
Given a graph G and a minimal density τ, the set of core nodes is defined by

$$coreNodes = \{v \in V \mid density(v) \geq \tau\}$$

Starting with a core node, we add further core nodes, which are in a certain range of the node, to our cluster to let it grow. In this step we have to take care of the underlying graph structure. Two aspects are important and have to be considered.

Aspect 1: We have to grow the clusters along our direct influence definition. A node only interacts with the nodes in its direct influence region; thus, if these nodes are core nodes they should belong to the same cluster. In Figure 2 a directed graph is given where the core nodes are marked with solid lines and the non-core nodes with dashed lines. For ease of presentation we do not show the densities or edge weights. If we select v_1, the nodes $\{v_3, v_4\} = direct(v_1) \cap coreNodes$ should correspond to the same dense region and hence to the same interesting subgraph. Because a directed graph is given, also the predecessors $\{s \in coreNodes \mid v_1 \in direct(s)\} = \{v_2\}$ have to be included. This is equivalent to $v_1 \in direct(v_2) \cap coreNodes$. The same procedure is now applied for v_2, v_3 and v_4, so that we add the nodes v_5 and v_6 to the cluster.

Definition 5. *Core of a cluster*
A non-empty subset $C \subseteq coreNodes$ is a core of a cluster iff

$$\forall\ v \in C : \forall\ s \in coreNodes : s \in direct(v) \vee v \in direct(s) \Rightarrow s \in C$$

and C is minimal among these sets.

By this definition, a cluster grows until for each node the required property is fulfilled. In Figure 2 the two minimal sets that fulfill this property are highlighted. These minimal sets are the dense areas that build our clusters cores. In Figure 3 we show examples for the identification of two different cores to point out the advantages of our definition. In Figure 3(a) we get two cores even though the two core nodes v_1 and v_2 are connected. The connection is only very loose; thus, both nodes do not influence each other. In Figure 3(b) we get two cores even if v_1 and v_2 are reachable over a path. This path, however, has to use the non-core node v_3. This non-core node indicates the existence of two different clusters.

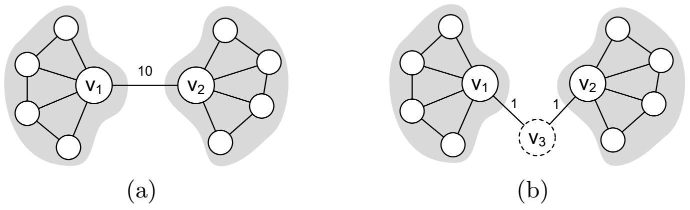


Fig. 3. Examples for the identification of two different cores

Aspect 2: The second aspect we have to consider is the non-symmetry of our direct influence region, i.e. we can have $d \in direct(s) \not\Leftrightarrow s \in direct(d)$. Why are directed graphs a particular challenge? For this, we first consider the Definition 5 for undirected graphs. Consequently with our model we get $d \in direct(s) \Leftrightarrow s \in direct(d)$ and Definition 5 simplifies to

$$\begin{aligned} &\forall\, v \in C : \forall\, s \in coreNodes : s \in direct(v) \Rightarrow s \in C \\ &\qquad\qquad\qquad\qquad \Leftrightarrow\ \forall v \in C : coreNodes \cap direct(v) \subseteq C \end{aligned}$$

for a minimal non-empty set C.

A useful property for interpreting the cores is the core path property. A path (within a cluster core) that uses only direct influence edges and that connects s with d is called a core path for s and d. Formally we define the existence of such a path with respect to a cluster core C and two nodes by:

$$\begin{aligned} &corePath_C(s,d) = TRUE \Leftrightarrow \exists v_1, \ldots, v_n \in C : \\ &\qquad\qquad v_1 = s \wedge v_n = d \wedge v_{i+1} \in direct(v_i) \wedge v_i \in coreNodes \end{aligned}$$

In Figure 4(a) the node set $C = \{v_1, \ldots, v_7\}$ is a valid cluster core. The nodes v_4 and v_7 for example are connected via a core path. The path $< v_4, v_1, v_5, v_7 >$ uses only nodes within C and each successor is in the direct influence region of its predecessor. If each pair of nodes within the core C is connected via a core path, C fulfills the core path property. In Figure 4(a) this property holds.

Definition 6. *Core path property*
The core C fulfills the core path property iff

$$\forall s, d \in C : corePath_C(s,d) = TRUE$$

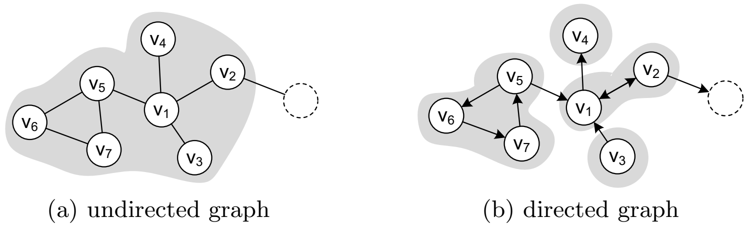


Fig. 4. Core path property and strong cluster cores

One can prove that a cluster core in an undirected graph always fulfills the core path property. Furthermore, a cluster core C is maximal with respect to the core path property, i.e. there exists no $C' \supset C$ that fulfills also the core path property. In Figure 4(a) we cannot add further nodes to C without violating the core path property. Thus, in an undirected graph the cluster cores correspond to the maximal sets that fulfill the core path property.

For a directed graph the Definition 5 and the maximal sets with respect to the core path property do not necessarily lead to the same results. In Figure 4(b) we get the cluster core $C = \{v_1, \ldots, v_7\}$. However, as one can see the node v_4 is not connected to v_7 via a core path; the nodes v_3 and v_5 are not connected at all. In directed graphs the core path property is a more restrictive criterion for a cluster. The Definition 5 is fulfilled by the nodes $\{v_1, \ldots, v_7\}$ in Figure 4(b) while the core path property e.g. only for the nodes $\{v_1, v_2\}$ or $\{v_5, v_6, v_7\}$. These sets are highlighted in Figure 4(b). Thus, for directed graphs we can define another stronger definition for a core of a cluster:

Definition 7. *Strong core of a cluster*
A non-empty subset $C \subseteq coreNodes$ is a strong core of a cluster iff C fulfills the core path property and C is maximal.

Obviously each strong core SC is a subset of a core C, i.e. $SC \subseteq C$. Thus, we have the restrictive strong core property, which yields small cluster cores, and the weaker core property, which yields larger clusters. We want to analyze a further version in between these extremes. In contrast to the strong core property, where each pair of nodes is reachable in both directions, we make a relaxation that only requires a core path in one direction.

Definition 8. *Semi-strong core of a cluster*
A non-empty subset $C \subseteq coreNodes$ is a semi-strong core of a cluster iff

$$\forall s, d \in C : corePath_C(s, d) = TRUE \ \vee corePath_C(d, s) = TRUE$$

and C is maximal.

In Figure 4(b) for example the node set $C = \{v_1, v_2, v_3, v_4\}$ forms a semi-strong core. The node v_3 can reach all nodes in C via a core path, v_2 the nodes $\{v_1, v_4\}$ and v_1 the nodes $\{v_2, v_4\}$. For each pair we get at least one core path in a single direction.

Precluster and postcluster. The cores are the most important nodes that form the clusters. Additionally, two other sets of nodes can be defined. Given a cluster core C, the densities of all nodes within the core exceed a certain threshold. Thus, an interesting node set contains all nodes that account for the density of the core, i.e. removing one of these nodes the densities of the core nodes change. We call this set a precluster. On the other hand we can also define the postcluster of C. This set contains all objects that are influenced by the core.

Definition 9. *Precluster and postcluster*
Given a cluster core C, the precluster $Pre(C)$ and postcluster $Post(C)$ contain the nodes

$$Pre(C) = \left(\bigcup_{d\in C} revRegion(d)\right) \setminus C \text{ and } Post(C) = \left(\bigcup_{d\in C} region(d)\right) \setminus C$$

2.4 Graph-Theoretic View and Algorithmic Aspects

In the following we want to point out a graph-theoretic view of our model that helps us to implement an efficient algorithm. We first transform our graph. We remove all non-core nodes and those edges that do not contribute to a direct influence. Only along the remaining nodes and edges a core could grow. Overall, this so called residual graph is defined by $V' = coreNodes$ and $E' = \{(s,d) \in E | d \in direct(s) \wedge \{s,d\} \subseteq coreNodes\}$. In Figure 5 we show an original graph; non-core nodes are highlighted with dashed lines. The two edges labeled with 'noise edge' should indicate, that v_3 and the other node are not directly influenced by v_2 even if they are connected. If we remove these two edges as well as the nodes n_1 and n_2, the residual graph on the right is obtained.

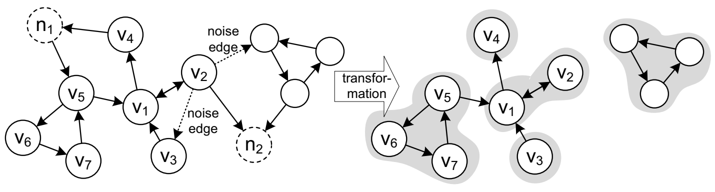


Fig. 5. Graph transformation to the residual graph

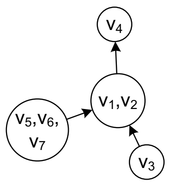


Fig. 6. Quotient graph

The weak components [22] of the residual graph are our cluster cores following Definition 5. The strong components [22] build our strong cluster cores following Definition 7. In Figure 5 (right) we highlighted these node sets. The semi-strong cores obey a more complex structure. Derived from the strong components we can construct a quotient graph. Each node in the quotient graph corresponds to a strong component (cf. Fig. 6). The nodes are connected with a directed edge if at least one edge in the original graph exists between these node sets. This quotient graph is a DAG (directed acyclic graph). Each maximal path from a root node to a leaf node is then a semi-strong component. In Figure 6 we get the two paths $< \{v_3\}, \{v_1, v_2\}, \{v_4\} >$ and $< \{v_5, v_6, v_7\}, \{v_1, v_2\}, \{v_4\} >$ that define the two semi-strong cores. Thus, the strong, semi-strong and weak components in the residual graph correspond to the definitions of the cluster cores. Due to space limitations we omit the proof.

A comparison of the three definitions yields an interesting conclusion. While the weak and strong components are disjoint sets, the semi-strong node sets can overlap. The cores of our clusters following the semi-strong definition are allowed to share some objects. This overlap or disjointness is already visible in our example in Figure 5 and 6 respectively.

Based on the graph-theoretic properties, our algorithm can efficiently determine the interesting subgraphs. We call our algorithm *GDens*, due to the density calculation in graphs. First, for our density calculation we have to determine the shortest path distances. We use an adaption of Dijkstra's algorithm. By this we can use an early stopping of the density calculation if the distances exceed the maximal distance h. We have to apply Dijkstra's algorithm for each node to determine the overall densities.

After the density calculation step we have to determine the cores of the clusters. For our weak core definition we have to identify the weak components in the residual graph; this can be done by a depth-first search procedure. The strong-components and hence the quotient graphs within each weak component are identified by Tarjan's algorithm. Finally, we generate the maximal paths within the quotient graphs to identify the semi-strong cores.

Summarized, our GDens utilizes efficient graph algorithm methods for identifying the interesting subgraphs. Our beforehand defined core definitions can be reduced to well known properties in the residual graph and hence emphasize the use of these cluster cores. In total, we get the possibility to flexibly identify and interpret different interesting subgraphs based on the density calculation technique in directed and weighted graphs.

2.5 Complexity Analysis

We briefly analyze the complexity of our algorithm with respect to a graph $G = (V, E, w)$. Let us assume that in average the influence is larger than zero for a fraction of x percent of all nodes. Based on the complexity of Dijkstra's algorithm and as we have to use Dijkstra's algorithm for each node to determine the densities, the overall complexity for the density computation step is

$$O\left(|V| \cdot [\ x \cdot |V| \cdot log(x \cdot |V|) + x \cdot |E|\ ]\right) = O(x \cdot |V|^2 \cdot log(x \cdot |V|) + x \cdot |V| \cdot |E|)$$

Obviously in the worst case the influence is larger than zero for all nodes, i.e. $x = 1$, and we have a dense graph, i.e. $O(|E|) = O(|V|^2)$. In this case we can infer the worst case complexity of

$$O\left(|V|^2 \cdot log|V| + V \cdot |V|^2\right) = O(|V|^3)$$

If we assume the positive influence of a node is restricted to a constant number of nodes, i.e. $x = O(1/|V|)$, and a sparse graph with e.g. $O(|E|) = O(c \cdot |V|)$ is given, we get a complexity of

$$O(1/|V| \cdot |V|^2 \cdot log(1/|V| \cdot |V|) + 1/|V| \cdot |V| \cdot c \cdot |V|) = O(c \cdot |V|)$$

In the second step, we have to determine the cores. The depth-first procedure for identifying the weak-components in the residual graph has a complexity of $O(|V| + |E|)$. Tarjan's algorithm for identifying the strong-components has a complexity of $O(V_{max} + E_{max})$, if V_{max} is the maximal number of nodes for all weak components and E_{max} is the maximal number of edges. For the semi-strong core definition we additionally have to generate the maximal paths within the quotient graph. Assuming that we identify k quotient graphs each with n strong components we get an additional complexity of $O(k \cdot e^{n/e})$ (proof skipped). In realistic scenarios we have $k, n \ll |V|$ and hence the term is almost negligible.

3 Experiments

In the next section we analyze the runtime and quality of our *GDens* algorithm.

Setup. We use two variants of our algorithm. *GDens (core)* uses the identified cluster cores as the interesting subgraphs. *GDens (all)* includes the precluster and postcluster to build the interesting subgraphs. For comparison we use the algorithms of [6], called *Voltage*, and [13], called *Edge Betweenness*. All input parameters are optimized. All implementations are in Java. For runtime and quality comparison we generate synthetic data following the methods in [13, 18] and adding uniformly distributed edge weights. In average, edge weights within clusters are two times smaller than edge weights between clusters. As not stated otherwise we hide 20 clusters with 10.000 nodes and 30.000 edges. Clustering quality is measured with the F1 score [23, 24]. The F1 value is the harmonic mean of precision and recall, i.e. an identified subgraph has to detect most of the nodes (recall) but also only the nodes (precision) of a hidden cluster. Keep in mind that for our algorithm two F1 scores can be calculated based on GDens (core) or GDens (all).

Cluster core definitions. In Figure 7 we analyze the effects of our different cluster core definitions. On the left the F1 value is illustrated. The solid bars correspond to the quality for GDens (all). The white bars inside correspond to the quality if only the core is considered, i.e. GDens (core) is used. The quality of GDens (core) decreases slightly with a more restrictive core definition. The

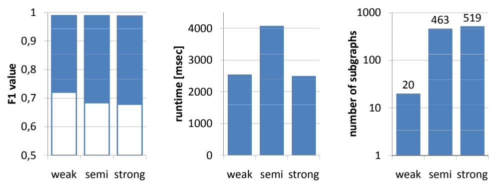


Fig. 7. Cluster core definitions

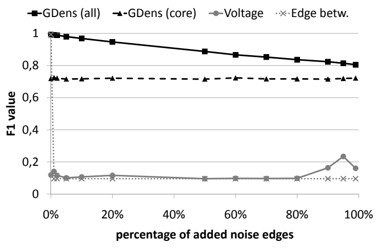


Fig. 8. Quality w.r.t. noise

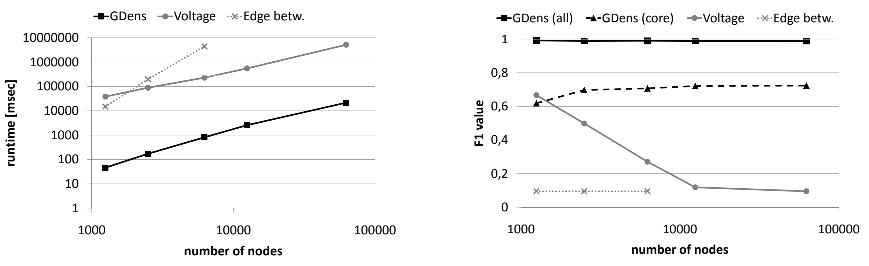


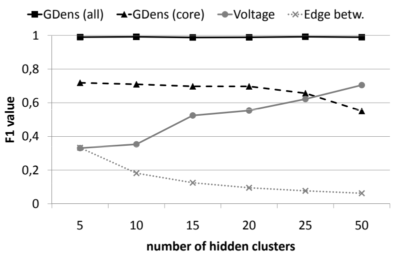
Fig. 9. Runtime and quality w.r.t. number of nodes

cores following the strong cluster core definition are very small and hence we cannot detect all nodes of the clusters. However, if we include the pre/postcluster the quality for all cluster core definitions is high. In the middle the runtime is analyzed. The weak and semi core definitions run in nearly equal time, while the semi-strong core needs more calculations. The determination of the paths within the quotient graph is a more complex task. On the right, the number of subgraphs in the mining result is printed. As expected the weak core definition yields the fewest clusters, while the other two definitions split the weak component in several smaller ones. In the following experiments we focus on the weak core definition as the quality is high and the runtime is low.

Noise. In Figure 8 we increase the number of noise edges in the graph, i.e. we add edges that do not belong to the communities. The higher the value the more difficult is the identification of interesting subgraphs. The quality of GDens (core) is not affected by adding noise; the dense areas are still identified. The high quality of GDens (all) decreases slightly because the pre/postcluster include more and more nodes that do not belong to the cluster. However, even for very high percentages of noise GDens shows in both variants high quality. The quality of the Voltage algorithm remains unchanged with very low quality results. Edge betw. reaches a very high quality for zero percentage of noise but then it rapidly decreases. Both algorithms cannot identify the true hidden clusters.

Number of nodes. In Figure 9 (left) we plot the runtime of the algorithms with respect to the number of nodes. Our GDens algorithm is far more efficient than the other approaches. The Edge betw. method did not even finish within 12 hours for a dataset with 12500 nodes. Additionally in Figure 9 (right) we analyze the quality. While the quality of GDens in both variants stays on a high level or even increases a bit, the quality of Voltage decreases. In large graphs this algorithm cannot identify good patterns. Edge betw. has always low quality.

Hidden clusters. In the next experiment we analyze the effects if the number of hidden clusters in the graph is altered. Due to the high runtime of Edge betw. the number of nodes is set to 2000. As depicted in Figure 10 (left), with increasing number of clusters the quality of GDens is not or only less affected. With increasing number of clusters and fixed number of nodes, the interesting subgraphs get smaller and hence their identification is harder. The Edge betw.



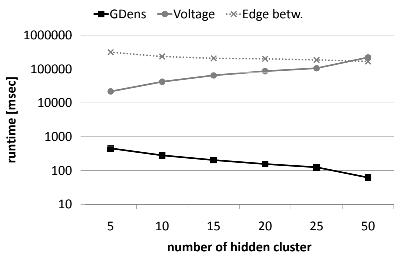


Fig. 10. Quality and runtime w.r.t. number of hidden cluster

algorithm shows a similar decrease but on a much lower quality level. Interestingly the quality of Voltage increases. The algorithm is better in the detection of small clusters. However, the quality of GDens (all) is never reached by this algorithm. Considering the runtime of the algorithms in Figure 10 (right), we see that the quality increase of Voltage is paid with a high and also increasing runtime. The runtime of GDens is orders of magnitudes lower. Furthermore, with more and hence smaller subgraphs the runtime even decreases.

Summarized, our GDens outperforms the competing algorithms in quality as well as in runtime. It is able to detect the interesting subgraphs also in noisy settings and automatically identifies the number of clusters.

Parameter variation. In our next experiment in Figure 11 we vary the parameter τ. On the right y-axis we indicate the number of identified clusters. First, the number increases because the hidden clusters are split-up in several smaller ones due to a higher τ. Afterwards, the number decreases to zero because no more core objects are identified. Correspondingly, the quality of the clustering (left y-axis) based on the core objects decreases. Considering GDens (all) the quality drops but at a later step. The objects that are included in the hidden clusters but not identified by the cores are now contained in the pre/postclusters.

DBLP data. To analyze real world data, we use the DBLP data set where nodes represent authors and edges/edge-weights the number of co-written papers. We generate a graph based on all publications from 2004-2008 and we extracted the largest connected component with 228k nodes and 1458k edges. In Figure 12 we

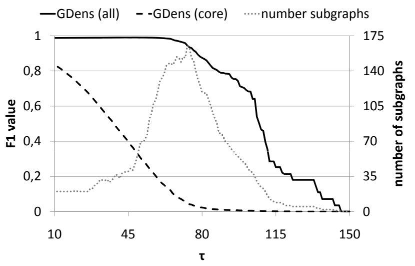


Fig. 11. Results for different τ values

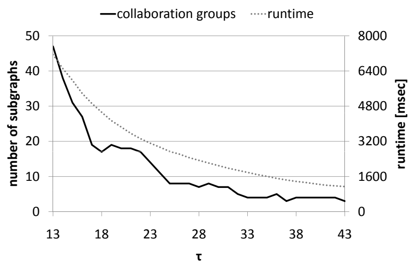


Fig. 12. Results on DBLP data

	GDens (core)			GDens (all)		
	weak	semi	strong	weak	semi	strong
weighted graph	0.72	0.68	0.68	0.99	0.98	0.98
unweighted graph	0.36	0.26	0.23	0.96	0.96	0.96

Fig. 13. Clustering quality (F1 value) for weighted vs. unweighted graphs

present some results with respect to a varying τ value. The F1 value cannot be determined for this data set because the true clustering structure is not known. Instead we plot the number of identified cluster cores with more than 5 nodes, i.e. these clusters correspond to large collaboration groups. Additionally, the runtime of the algorithm is presented. Both measures decrease continuously, i.e. we identify less collaboration groups but increase the efficiency. We want to point out the high efficiency of our GDens algorithm also on this large DBLP data set.

Unweighted graphs. In the next experiment we want to focus on the advantage of our algorithm to handle weighted graphs. In Figure 13 we show the difference in clustering quality if instead of the weighted graph an unweighted one is used, i.e. we ignore the weights by setting these to a constant value. As one can see in all cases the quality of the unweighted clustering is smaller. Especially if we only consider the core of the cluster (middle column) the quality decreases. This experiment supports the need for interesting subgraph mining algorithms that incorporate the weights of edges as our model does.

4 Conclusion

We introduce a novel technique to identify interesting subgraphs using the method of influence functions for calculating the densities of nodes. Our model can handle directed and weighted graphs and we show in experiments that using this information increases the quality of the clustering result. We present three types of dense subgraphs that account for the direction of edges in the graph. Our GDens algorithm identifies the number of clusters automatically and it is robust with respect to noise. In experiments we demonstrate the high quality and low runtime of our GDens algorithm compared to other subgraph mining methods.

Acknowledgment

This work has been supported by the UMIC Research Centre, RWTH Aachen University, Germany.

References

1. Feige, U., Peleg, D., Kortsarz, G.: The dense k-subgraph problem. Algorithmica 29(3), 410–421 (2001)
2. Abello, J., Resende, M.G.C., Sudarsky, S.: Massive quasi-clique detection. In: Rajsbaum, S. (ed.) LATIN 2002. LNCS, vol. 2286, pp. 598–612. Springer, Heidelberg (2002)

3. Liu, G., Wong, L.: Effective pruning techniques for mining quasi-cliques. In: ECML/PKDD, vol. (2), pp. 33–49 (2008)
4. Flake, G.W., Lawrence, S., Giles, C.L.: Efficient identification of web communities. In: KDD, pp. 150–160 (2000)
5. Long, B., Wu, X., Zhang, Z.M., Yu, P.S.: Unsupervised learning on k-partite graphs. In: KDD, pp. 317–326 (2006)
6. Wu, F., Huberman, B.A.: Finding communities in linear time: A physics approach. CoRR cond-mat/0310600 (2003)
7. Ruan, J., Zhang, W.: An efficient spectral algorithm for network community discovery and its applications to biological and social networks. In: ICDM, pp. 643–648 (2007)
8. Long, B., Zhang, Z.M., Yu, P.S.: A probabilistic framework for relational clustering. In: KDD, pp. 470–479 (2007)
9. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.J.: Scan: a structural clustering algorithm for networks. In: KDD, pp. 824–833 (2007)
10. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE TPAMI 22(8), 888–905 (2000)
11. Meila, M., Pentney, W.: Clustering by weighted cuts in directed graphs. In: SDM (2007)
12. Ding, C.H.Q., He, X., Zha, H., Gu, M., Simon, H.D.: A min-max cut algorithm for graph partitioning and data clustering. In: ICDM, pp. 107–114 (2001)
13. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA 99, 8271–8276 (2002)
14. Gregory, S.: An algorithm to find overlapping community structure in networks. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 91–102. Springer, Heidelberg (2007)
15. Gregory, S.: A fast algorithm to find overlapping communities in networks. In: ECML/PKDD, vol. (1), pp. 408–423 (2008)
16. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Phys. Rev. E 70, 66111 (2004)
17. Newman, M.E.J.: Modularity and community structure in networks. PNAS USA 103, 8577–8582 (2006)
18. Chen, J., Zaïane, O.R., Goebel, R.: Detecting communities in social networks using max-min modularity. In: SDM, pp. 978–989 (2009)
19. Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapman and Hall, London (1986)
20. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases. In: KDD, pp. 226–231 (1996)
21. Kailing, K., Kriegel, H.P., Kröger, P.: Density-connected subspace clustering for high-dimensional data. In: SDM, pp. 246–257 (2004)
22. Godsil, C., Royle, G.: Algebraic Graph Theory. Springer, Heidelberg (2001)
23. Müller, E., Günnemann, S., Assent, I., Seidl, T.: Evaluating clustering in subspace projections of high dimensional data. PVLDB 2(1), 1270–1281 (2009)
24. Van Rijsbergen, C.J.: Information Retrieval. Butterworths, London (1979)

Finding Itemset-Sharing Patterns in a Large Itemset-Associated Graph

Mutsumi Fukuzaki[1], Mio Seki[1], Hisashi Kashima[2], and Jun Sese[1]

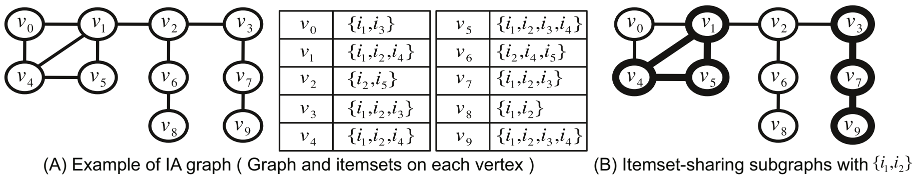
[1] Dept. of Computer Science, Ochanomizu Univ. 2-1-1 Otsuka, Bunkyo, Tokyo, Japan
[2] Dept. of Math. Informatics, Univ. of Tokyo. 7-3-1 Hongo, Bunkyo, Tokyo, Japan

Abstract. Itemset mining and graph mining have attracted considerable attention in the field of data mining, since they have many important applications in various areas such as biology, marketing, and social network analysis. However, most existing studies focus only on either itemset mining or graph mining, and only a few studies have addressed a combination of both. In this paper, we introduce a new problem which we call *itemset-sharing subgraph (ISS) set enumeration*, where the task is to find sets of subgraphs with common itemsets in a large graph in which each vertex has an associated itemset. The problem has various interesting potential applications such as in side-effect analysis in drug discovery and the analysis of the influence of word-of-mouth communication in marketing in social networks. We propose an efficient algorithm *ROBIN* for finding ISS sets in such graph; this algorithm enumerates connected subgraphs having common itemsets and finds their combinations. Experiments using a synthetic network verify that our method can efficiently process networks with more than one million edges. Experiments using a real biological network show that our algorithm can find biologically interesting patterns. We also apply ROBIN to a citation network and find successful collaborative research works.

1 Introduction

Since the origin of the field of data mining, frequent pattern mining has been one of the main topics of interests for researchers in this field. These researchers initially worked on itemset patterns [1, 2] in the context of market basket analysis; these studies were later extended to event sequence patterns [3]. Recently, graph-structured data have attracted considerable attention [4–6] because graph pattern mining can be applied to many interesting application areas such as biological networks, social networks, and the Web. While itemset mining seeks frequent combinations of items in a set of tuples, graph mining seeks frequent subgraphs in a set of graphs. Most of the prior studies have addressed only one type of their patterns, and only a few studies have considered combinatorial mining of two types of data structures [7–9].

In this paper, we consider a new combinatorial mining problem of itemsets and subgraphs, which we call the *itemset-sharing subgraph (ISS) set* enumeration problem. Let us assume that we have a graph in which each vertex is associated with an itemset (Fig. 1(A)). We refer to this graph as an *itemset-associated graph.* Our


v_0
v_1
v_2
v_3
v_4
v_5
v_6
v_7
v_8
v_9
(A) Example of IA graph (Graph and itemsets on each vertex)
(B) Itemset-sharing subgraphs with $\{i_1,i_2\}$

v_0	$\{i_1,i_3\}$	v_5	$\{i_1,i_2,i_3,i_4\}$
v_1	$\{i_1,i_2,i_4\}$	v_6	$\{i_2,i_4,i_5\}$
v_2	$\{i_2,i_5\}$	v_7	$\{i_1,i_2,i_3\}$
v_3	$\{i_1,i_2,i_3\}$	v_8	$\{i_1,i_2\}$
v_4	$\{i_1,i_2,i_4\}$	v_9	$\{i_1,i_2,i_3,i_4\}$

Fig. 1. An example of itemset-sharing subgraph

task is to enumerate the patterns that we call the ISS set, which is a set of large subgraphs in which all vertices share a large common itemset. The ISS set shown in Fig. 1(B) consists of two subgraphs depicted by the bold lines. All vertices in the ISS set share the itemset $\{i_1, i_2\}$. Although the subgraph consisting of only v_8 also shares the itemset, it is not included in the ISS set since it is quite small. Similarly, although the subgraph consisting of $\{v_0, v_1, v_4, v_5\}$ shares $\{i_1\}$, it is not sufficient to be an ISS set since the shared itemset is very small. The ISS set enumeration problem differs from other graph mining problems in the sense that the subgraphs included in an ISS set need not be identical.

We now illustrate how ISS sets are used in drug discovery. Let us consider the metabolic pathway networks, which describe biochemical processes occurring within a cell. A pathway is represented as a graph, where vertices denote genes and chemical compounds and edges denote chemical reactions among the genes and the compounds. The pathway networks play a considerably important role in drug discovery, because by finding a sub-pathway that is closely related to a disease, we can determine the target genes or chemical compounds on which the drug candidate should act. However, the drug candidate can affect several different pathways simultaneously, which may lead to unexpected outcomes. Such phenomena are called side effects. We would like to not only find the drug targets but also predict the side effects that may be caused by the action of the drug on the targets. Taking the drugs into account, we considered a pathway network to be an itemset-associated graph (Fig. 1(A)), where each vertex (a gene or a chemical compound) is associated with an itemset that indicates the set of drugs activating the gene or the compound ($\{i_1, i_2, \ldots, i_5\}$ shown in Fig. 1(A)). In the above context, an ISS set (Fig. 1(B)) corresponds to a set of sub-pathways that share the common activation drug; this implies that there are hidden or unknown connections among the sub-pathways and that the drugs designed to target genes or compounds in one sub-pathway might also act on the other sub-pathways. In Fig. 1(B), the sub-pathway consisting of only v_8 is also activated by the drugs $\{i_1, i_2\}$. However, this sub-pathway is very small, which implies that the activation would result from accidental observations. Large sub-pathways are more reliable and indicate that the side effects are more serious because these effects cover a wide range of pathway networks. Similarly, we expect that as the size of the set of common activation drugs increases, the possibility of the occurrence of side effects increases. Therefore, networks that consist of the large sub-pathways with a large set of activation drugs are important clues in predicting side effects for drug discovery and biological experimental design.

Let us now consider a marketing scenario in social network analysis. In social networks, vertices are considered to be participants, and edges are considered to be the relationships between these participants (e.g., friendships). Let us assume that each participant (vertex) is associated with the items that he or she has bought. The network can be considered to be an itemset-associated graph, and the subgraphs with large common itemsets can be regarded as underlying communities. Further, in a social network, common itemsets shared by many communities are considered as the (sets of) products that can be easily marketed through word-of-mouth communication; hence, the products with common features would be suitable for social marketing.

In order to solve the ISS set enumeration problem, one approach is to use an itemset mining technique [1, 2] to obtain all the frequent itemsets and then checking the connections between the itemsets in the networks. However, the itemset mining in real dataset are very long computation time because the supports (frequencies) of the items included in the ISS sets are usually low. To overcome the computation time problem, we propose an efficient algorithm called *ROBIN*. The ROBIN algorithm consists of two stages; it enumerates subgraphs that are larger than a specified threshold value at the first stage, and then it combines them at the second stage. By introducing effective pruning techniques in both the stages, we can enumerate the graphs very efficiently.

Finally, the efficiency of the proposed algorithm is shown in the experiments by using a synthetic dataset. ROBIN can solve problems with more than 100K vertices and $1,000$K edges for about a half hour. In the experiments using a real biological network, we discover hidden connections in metabolic pathways; this suggests the practical utility of ROBIN in the context of drug discovery. Furthermore, by applying ROBIN to a citation network, we find interesting patterns indicating successful collaborative works containing well-known database research topics. In both of the real dataset experiments, we show that execution time of ROBIN are faster than that of the method which first enumerates the itemset and then checks their connectivity.

Our contributions are summarized as follows:

1. We introduce the ISS set enumeration problem, which has sound potential applications in various real-world problems including finding side effects in drug discovery and estimating effects of word-of-mouth communication in marketing in social networks.
2. We propose a very efficient algorithm called ROBIN to solve the ISS set enumeration problem; In the ISS enumeration stage, we develop a novel pruning technique using hash tables called a *visited itemset table*, which stores itemsets shared by generated subgraphs, and allows us to enumerate ISSes very efficiently. In the ISS combination stage, we propose an efficient algorithm for finding ISS sets using an extend depth-first search (DFS) tree called *ISS tree*, which enables us to generate combinations of ISSes without unnecessary checking of graph inclusion.
3. We conduct experiments using two real-world network data, a biological network and a citation network, and show scenarios where the ISS set

enumeration problem is useful. The results also show that ROBIN is much faster than the itemset-enumeration approach.

2 Itemset-Sharing Subgraph (ISS) Set Enumeration Problem

In this section, we introduce a novel data mining problem for analyzing itemset-associated graphs, which we refer to as the ISS set enumeration problem.

Let G be an undirected[1], unlabelled, and unweighted graph with an itemset on each vertex. We refer to this graph as an *itemset-associated graph (IA graph)*. Let $V(G)$, $E(G)$ and $\mathcal{I}(G)$ respectively signify a set of the vertices in G, a set of edges in G and a set of itemsets on vertices in G. Note that the size of graph G is given as the number of edges, i.e., $|G| = |E(G)|$.

We next define subgraphs whose vertices share itemsets.

Definition 1. *(Shared Itemset) Let G' be a connected subgraph of an IA graph G, where G' is also an IA graph. We define $I(G')$ as $I(G') = \bigcap_{v \in V(G')} I(v)$, and refer to $I(G')$ as a shared itemset of G'.*

Among the subgraphs having a shared itemset, we focus on an important subset, which cannot be expanded while retaining the currently shared itemsets.

Definition 2. *(Itemset-Sharing Subgraph (ISS)) We call G' an itemset-sharing subgraph (ISS) with $I(G')$ if $I(G') \neq \phi$ and $I(v) \not\supseteq I(G')$ for any vertex v in the neighbor vertices of G'.*

Note that the itemset shared by an ISS is defined without reference to its edges.

Now, we define the sets of ISSes that we want to enumerate in our task. As described in Section 1, sets of ISSes are useful in the context of drug discovery and marketing in social networks.

Definition 3. *(ISS Set) Let $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ be a set of ISSes, where each G_i is an ISS. Define $I(\mathcal{G})$ as $I(\mathcal{G})=\bigcap_{G \in \mathcal{G}} I(G)$. Note that $I(\mathcal{G})=\bigcap_{G \in \mathcal{G}} \bigcap_{v \in G} I(v)$. We call $\mathcal{G}$ an ISS set with $I(\mathcal{G})$, if all of the following conditions are satisfied: (1) $V(G_i) \cap V(G_j) = \phi$ for any G_i and G_j $(i \neq j)$ in $\mathcal{G}$. (2) $I(v) \not\supseteq I(\mathcal{G})$ for any vertex v in the neighbor vertices of $G' \in \mathcal{G}$. (3) $|G_i| \geq \theta_S$, where θ_S is a user-specified value. (4) No ISS G' with $I(\mathcal{G})$ exists except in $\mathcal{G}$.*

The first two conditions are an extension of the definition of ISS for dealing with multiple ISSes. The third condition gives the minimum size of the obtained ISSes because larger ISSes are of greater interest to us. The last condition ensures the maximality of the found ISS sets. Let $|\mathcal{G}|$ indicate the number of disconnected components of G, and hence, $|\mathcal{G}| = n$.

Finally, we define our new data mining problem where the task is to enumerate all ISS sets from a given IA graph.

[1] Although for simplicity, we assume that G is undirected, our method can be used for directed graphs in the same manner.

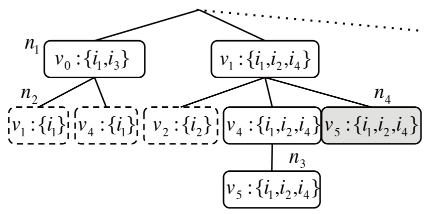
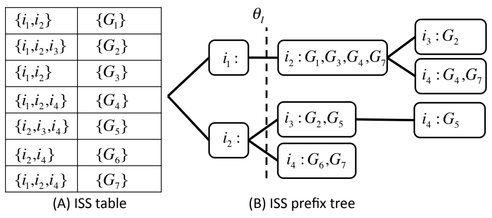


Fig. 2. The DFS itemset tree for Fig. 1(A)



$\{i_1,i_2\}$	$\{G_1\}$
$\{i_1,i_2,i_3\}$	$\{G_2\}$
$\{i_1,i_2\}$	$\{G_3\}$
$\{i_1,i_2,i_4\}$	$\{G_4\}$
$\{i_2,i_3,i_4\}$	$\{G_5\}$
$\{i_2,i_4\}$	$\{G_6\}$
$\{i_1,i_2,i_4\}$	$\{G_7\}$

Fig. 3. An ISS table and an itemset-graph prefix tree

Definition 4. *(ISS Set Enumeration Problem) Given an IA graph and user-specified values θ_S, θ_I and θ_F, from the IA graph, enumerate all ISS sets* $\mathbf{G}$ *satisfying $|\mathcal{G}| \geq \theta_F$, $|I(\mathcal{G})| \geq \theta_I$ for any ISS set $\mathcal{G} \in$* $\mathbf{G}$, *and $|G| \geq \theta_S$ for any ISS $G \in \mathcal{G}$ in any ISS set $\mathcal{G} \in$* $\mathbf{G}$.

3 Proposed Method

In this section, we propose an efficient algorithm called *ROBIN* (RelatiOn Between Items and Networks) for solving the ISS set enumeration problem. To solve the problem, one strategy first enumerates all the itemsets such as Apriori [1] and FP-trees [2], and then check the connectivity between the itemsets. The other strategy first enumerates the subgraphs, and then check the conditions of the subgraphs. Here, we use the latter method. We will show that the computing time of the former method requires longer than the latter method using real dataset in Section 4.

Robin consists of two stages. In the first stage, we enumerate all the ISSes efficiently by introducing *DFS itemset tree* and *visited itemset table*. Their details are described in Section 3.1. In the second stage, we generate ISS sets by combining the ISSes according to Section 3.2. In order to enumerate the ISS sets efficiently, we introduce an *ISS prefix tree* that contains the prefix of itemsets and their associated ISSes.

3.1 ISS Enumeration

In the first stage of the ROBIN, we enumerate ISSes from the given IA graph. We introduce efficient techniques for the enumeration of ISSes in this section. In the second stage of ROBIN, the obtained ISSes are combined with the generated ISS sets (Section 3.2).

We use a depth-first search (DFS) tree for enumerating ISSes $\mathcal{G}$ where $|G| \geq \theta_S$ and $|I(G)| \geq \theta_I$ for $G \in \mathcal{G}$. Each node of the tree contains a vertex and an itemset related to the path from the root to the node. We denote the tree as a *DFS itemset tree*. On the DFS itemset tree, we do not need to maintain edges because $I(G)$ can be computed from vertices and their itemsets.

The generation of the subgraphs itself is considered to be a simplified version of the DFS lexicographic order used in the gSpan algorithm [6], and hence, this

DFS itemset tree can avoid duplicate generation of identical graphs. Fig. 2 shows the DFS itemset tree for the IA graph in Fig. 1(A). Each node in the DFS itemset tree contains a vertex and an itemset. The vertices included in the path from the root to the tree node represent the vertices of the subgraph.

Thanks to the following monotonic property of ISS about itemset size, we can prune subtrees in the DFS itemset tree, which dramatically reduces the search space.

Property 1. *Let us denote two ISSes by G' and G'', and let $V(G') \supset V(G'')$. Then, $I(G') \subseteq I(G'')$ holds.*

The tree nodes indicated by dotted boxes in Fig. 2 can be pruned by using this property when $\theta_I = 2$.

The next theorem allows us to avoid generating subgraphs that have the same vertices as those of already generated subgraphs and have itemsets that are subsets of itemsets associated with the already generated graphs.

Theorem 1. *Let n_1 and n_2 be a pair of nodes of the DFS itemset tree, where n_1 was generated before n_2. If vertices associated with n_1 and n_2 are identical, and $I(n_1) \supseteq I(n_2)$, no ISS exists in a descendant of n_2.*

This theorem implies that if we visit one of already visited vertices and the common itemset of the current path is identical to or a subset of one of the itemsets of the previously visited vertices, we can prune the subtree rooted by the current node in the DFS itemset tree. Therefore, this property is useful for avoiding unnecessary exploration of subgraphs.

Theorem 1 prompts us to make the hash table from nodes to their related itemsets for efficient pruning of subgraphs. We call the hash table a *visited itemset table* and build it while constructing a DFS itemset tree.

Using the DFS itemset tree, we can generate all ISSes whose subgraph size is greater than θ_S and common itemset size is greater than θ_I. Fig. 3(A) illustrates the ISSes and their associated itemsets. We refer to this table as the *ISS table.* In the next section, in order to enumerate ISS sets efficiently, we introduce an efficient method of generating combinations of the ISSes.

3.2 ISS Set Combination

In this section, we introduce an efficient method for enumerating ISS sets from the ISS table created in the previous section. One simple method for computing the ISS sets is to generate combinations of all the ISSes. However, this procedure is quite redundant because different combinations of ISSes may result in the same shared itemset. Our method generates ISS sets efficiently by grouping ISSes by shared itemsets. Once we fix one itemset, an ISS set sharing the itemset is uniquely determined. Therefore, one approach to enumerating all ISS sets is to generate all itemsets that can be associated with ISS sets. For the efficient generation of the itemsets, we use the depth first search.

Definition 5. *(ISS Tree) Let T_I be a tree, each of whose node n contains itemset $I(n)$ and a set of ISSes $\mathcal{G}(n)$ which shares $I(n)$. The root of T_I contains an* *itemset including all items and a vacant set of ISSes. Let* n_1 *and* n_2 *be a pair of nodes of* T_I*. When* n_1 *is an ascendant of* n_2*,* $I(n_1) \supseteq I(n_2)$*.*

We call the tree *ISS tree*. Nodes closer to the root contain larger itemset. A child of a node in the ISS tree can be generated by adding an ISS to a parent node's ISSes, and its shared itemset can be computed. Thanks to the monotonic property of itemset size, we can prune subtrees in the ISS tree.

Although we can enumerate all the combinations of ISSes by the simple DFS method, the size of ISS tree may increase considerably especially when the number of ISSes is large. In order to efficiently generate the ISS tree, we add a group of ISSes sharing an itemset to ISSes in its parent node.

We here divide ISS sets into two types: explicit ISS sets and implicit ISS sets. Explicit ISS sets are associated with itemset appeared in an ISS table, while implicit ISS sets are associated with itemset which is a subset of itemsets appeared in the ISS table. We first generate explicit ISS sets quickly using prefix tree structure, and then produce implicit ISS sets by the combinations of explicit ISS sets.

Definition 6. *(Explicit and Implicit ISS Set) Let* $\mathcal{G}$ *be all the ISSes in an ISS table, and* $\mathcal{I}(\mathcal{G})$ *be itemsets associated with ISSes in* $\mathcal{G}$*. Let* $\mathcal{G}_I$ *be an ISS set with* I*. When* $I \in \mathcal{I}(\mathcal{G})$*, we call* $\mathcal{G}_I$ *an explicit ISS set; otherwise we call* $\mathcal{G}_I$ *an implicit ISS set.*

Basis of the above definition, any ISS set can be classified as explicit or implicit.

For the efficient generation of all ISS sets, we first extract all of the explicit ISS sets, and then generate ISS sets by removing overlapping ISSes. The following theorem guarantees us to generate all the ISS sets.

Theorem 2. *Let* $\mathcal{G}_I$ *be* $\mathcal{G}_C - \{G \mid G \subseteq G' \text{ where } G, G' \in \mathcal{G}_c\}$*. Then,* $\mathcal{G}_I$ *is an ISS set with* I*.*

This theorem allows us to generate the explicit ISS sets with I. All the explicit ISS sets can be generated by computing $\mathcal{G}_I$ for all the itemsets in the ISS table. However, the procedure requires many checks related to the inclusion relations among graphs. Here, we introduce an efficient way to generate explicit ISS sets by using a prefix tree representing itemsets.

Definition 7. *(ISS Prefix Tree) Let* T_P *be a tree, each of whose nodes* n *contains an item* i_n *and an ISS set* $\mathcal{G}(n)$*. Let denote two nodes in* T_P *by* n_1 *and where* n_1 *is an ascendant of* n_2*. Then,* $i_{n_1} < i_{n_2}$ *holds. Any itemset* I *in the ISS table is represented by a path in* T_P*. The ISS set in node* n *in* T_P *shares an itemset represented by a corresponding path from the root to* n*.*

We call the tree an *ISS prefix tree*. Using the ISS prefix tree, we represent all the associations between the itemsets contained in the ISS table and the ISSes. Fig. 3(B) represents the ISS prefix tree of Fig. 3(A). We put no ISSes to nodes whose depth is less than θ_I because none of the nodes generate ISS sets. Thanks to this prefix tree structure, we can accelerate the finding of the associations between itemsets and explicit ISS sets.

Table 1. Parameters for ROBIN

Name	Description	Default
	Parameters for ISS sets	
$\|I\|$	Size of the itemset shared by an ISS set	10
S	ISS size	7
F	Size of ISS set	5
P	Number of ISS sets patterns	10
Q	Number of ISSes not included in ISS sets	$\|V\|/30$

Name	Description	Default
	Parameters for graphs and itemsets	
$\|V\|$	Number of vertices	15,000
$\|E\|$	Number of edges	$10 \times \|V\|$
N	Number of items	100
$\|T\|$	Avg. size of itemsets in a vertex	10
	User-specified thresholds	
θ_S	Minimum ISS size	$S-1$
θ_I	Minimum shared itemset size	$\|I\|-1$
θ_F	Minimum size of ISS set	$P-1$

We here generate itemsets shared by implicit ISS sets by using the combination of two explicit ISS sets. From the itemset, we generate ISS sets by using the ISS prefix tree. The following theorem guarantees that the combinations can enumerate all of the implicit ISS sets.

Theorem 3. *Any itemset shared by an implicit ISS set is represented by the intersection of the itemsets shared by some of the explicit ISS sets.*

On the basis of this theorem, we can generate implicit ISS sets by using combinations of the itemsets shared by explicit ISS sets. Therefore, we generate a DFS tree each of whose nodes contains an itemset and an ISS set. We can prune the branches in the ISS tree from the monotonic property in Definition 5. Furthermore, the following property substantially reduces the search space.

Property 2. *Let node n contain an itemset $I(n)$ and an ISS set $\mathcal{G}(n)$. If $I(n)$ and an itemset I' of an existing node are identical, we need not traverse the branch rooted by n.*

To use these pruning techniques, we need not calculate inclusion relations between graphs in ISSes.

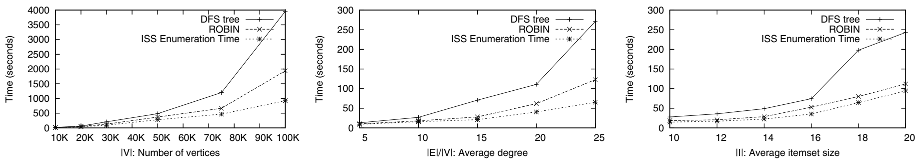
4 Experiments

In this section, we present the results of our experiments using a synthetic dataset and two real-world datasets.

4.1 Results for a Synthetic Network

We generated a synthetic network dataset in order to evaluate the performance of the ROBIN algorithm. The parameters for ROBIN and their default values are presented in Table 1. We generated synthetic datasets having $|V|$ vertices and $|E|$ edges. Each dataset includes P ISS sets whose shared itemset size is $|I|$ and size of ISS set is F. Moreover, we add the fake itemsets whose size is $1.7 \times |I|$. The detail procedure is omitted due to the space limitation. All experiments were performed using a 3.2 GHz AMD® Opteron™ machine with 1 GB memory running on Linux kernel 2.6. We implemented ROBIN in Java™ 5.

We investigated the efficiency of the ROBIN algorithm by using the synthetic network data and varying the size of the network, the average size of itemsets, and the parameters for ROBIN.



(A) Graph size (B) Average degree (C) Average itemset size

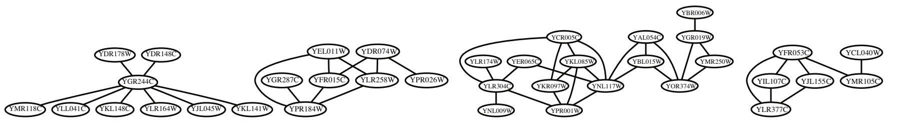
Fig. 4. Performance study with respect to overall graph size

In order to investigate the efficiency of enumerating combinations of the ISSes in ROBIN, we measured the execution times in the case of ROBIN (labeled as "ROBIN") and the times in the case of the algorithm in which we replace the ISS tree and the groups of ISSes generated by ISS prefix tree with the standard DFS tree by adding single ISS to its parent node to generate combinations of ISSes (labeled as "DFS tree"). We also show the execution times required for enumerating ISSes (Section 3.1) because these times are independent of the approach we choose. The differences between the execution times of ROBIN and ISS enumeration and those of the DFS tree approach and ISS enumeration indicate the computing time required to enumerate the combinations of ISSes.

Fig. 4(A) presents the execution times by varying the number of nodes in the network. This figure depicts that our method is more scalable than the alternative approach. The largest network in this experiment has 100K vertices and one million edges. The execution times increase quadratically with respect to the increase in the number of vertices. In particular, the larger the graph becomes, the larger is the execution time difference between the two approaches. Because the support (ratio of the number of vertices in ISS sets to the total number of vertices) was $F \times S/|V| = 0.0023$, when the values were set to the default values, it is difficult to find itemset patterns using the Apriori algorithm [1] and the FP-trees [2]. In contrast, ROBIN can work with such a low support and can still find important itemsets because it uses subgraphs that connect the itemsets.

Fig. 4(B) shows the execution times by varying the number of degrees in the network. In general, the execution time increases rapidly according to the density of the graph, because we need to check many neighbor vertices. However, our result demonstrates that the execution time of ROBIN increases rather gradually. We can observe that as the degree of the graph increases, the difference between the execution times of the two methods increases. This observation verifies the computational efficiency of ROBIN.

Next, we investigate the performance of ROBIN by varying the itemset size shared in ISSes. The dependence of the execution time on the itemset size is shown in Fig. 4(C). As shown in the figure, the average itemset size is not significant impact to ROBIN. Note that the algorithms succeeded in finding ISS sets with relatively large itemsets (more than 10 items). This result is in contrast to that of the existing studies on mining long patterns [10], in which finding low-frequency itemset patterns efficiently is difficult.



(A) TCA Cycle (B) Starch and sucrose metabolism (C) TCA Cycle and Pyruvate metabolism (D) Fructose and mannnose metabolism

Fig. 5. One of the ISS sets found in the real biological network. All of the vertices sharing five stress conditions.

Table 2. All of the ISS sets found in the DBLP citation network

No.	Authors	# of ISSes	# of refs.	# of papers
1	Rajeev Rastogi, Abraham Silberschatz	3	30	23
2	Amr El Abbadi, Divyakant Agrawal	2	40	25
3	Riccardo Torlone, Paolo Atzeni	2	13	11
4	Marc Gyssens, Dirk Van Gucht	2	12	11
5	Ling Liu, Calton Pu	2	11	11
6	Raghu Ramakrishnan, Praveen Seshadri	2	11	11

4.2 Results for a Biological Network

We applied ROBIN to a real metabolic pathway dataset with 6,152 vertices and 3,318 edges; here, the vertices and edges represent genes and chemical interactions, respectively. The dataset was obtained under 173 different stressed conditions [11] by using yeast microarrays. Each of the conditions causes stimuli to cells, and finding stimuli associated with treatments of diseases is a good starting point for development of new drugs. Therefore, we used the set of the conditions as the items. In biological systems, highly expressed genes play an important role within the cells. Therefore, we converted the quantitative values into Boolean values using a threshold t. We set the parameters as $t = 1.5$, $\theta_S = 7$, $\theta_I = 5$ and $\theta_F = 4$. The average itemset size in the dataset was 4.78, and its execution time was 35.9 seconds. We extracted eight ISS sets in total. One of the ISS sets depicted in Fig. 5 was associated with the conditions of 8 hours, 10 hours, 1 day, 2 days and 3 days grown under YPD condition at 30 degree Celsius; all of these conditions are high-nutrition and high-temperature conditions. Consequently, our algorithm could extract biologically consistent conditions automatically. The four connected graphs were associated with four biological metabolic pathways. Some genes in Fig. 5(C) are known as the activator of the TCA cycle including Fig. 5(A). Also associated pathways with Fig 5(D) are related to TCA cycle, and hence, the relationship between these two ISS sets is biologically reasonable.

4.3 Results for a Citation Network

We applied ROBIN to a citation network consisting of academic papers to demonstrate that ROBIN can extract successful collaborative researches automatically.

We create a citation network from the DBLP dataset [12], which is a snapshot of the DBLP as of April 12, 2006. Each vertex in the network corresponds to a paper and is associated with an itemset representing the author of the paper. Each edge indicates a citation. The DBLP network has $22,178$ vertices (papers), $112,304$ edges (citations), and $16,638$ items (authors). All papers have at least one author and one reference. The average number of authors for a paper is 2.29. We set the parameters for ROBIN as $\theta_I = 2$, $\theta_S = 10$, and $\theta_F = 2$.

Table 2 summarizes the six ISS sets found by ROBIN. The columns represent the ISS sets number, co-authors, numbers of disconnected networks, number of references in the ISS set, and number of papers in the ISSes. For example, the ISS set No.1 consists of three different ISSes, and the ISS set contains 23 papers and 30 references.

The research topics corresponding to the three ISSes in the ISS set No.1 are *multi-databases*, *video-on-demand storage*, and *main memory databases*. This result implies that Rajeev Rastogi and Abraham Silberschatz have successfully collaborated on three different research topics.

5 Related Work

In recent years, graph mining has received increasing interest from researchers. Frequent subgraph discovery methods [4–6, 13] enable us to enumerate all frequent common-structured subgraphs in a graph database. In this study, we are not concerned about the structure of the subgraph, and the existing methods cannot handle itemsets on subgraphs, hence we cannot apply the existing methods to our problem directly.

For the discovery of ISS sets, one straightforward approach might be to use the frequent pattern or closed itemset mining methods [1, 2, 14, 15] and then to check the connection among the found itemsets in the networks. However, in Section 4, we demonstrated that this approach is not efficient and requires huge amount of memory, which implies the effectiveness of ROBIN's approach which enumerates all subgraphs first.

The combinatorial mining of networks with numerical vectors has been studied in constrained clustering [7, 9]. The studies attempt to find the simultaneous clustering of the vertices in a network and the numerical vectors associated with the vertices. One significant difference between our problem and these problems is that the associated features on every vertex are discrete values in our problem. This property makes it difficult to apply the constrained clustering methods to our problem. MATISSE [16] and CoPaM [17] study the combinatorial mining of networks with feature vectors. Both methods find dense subgraphs whose vertices having similar features. However, we are not concerned about the density of the subgraph, and our method can find the *sparse* hub network shown in Fig. 5(A). Hashimoto *et al.* [8] proposed a combinatorial mining of sequence structured data and tree structured data. Their approach can be naturally extended to handle graph structured data, but the goal of our problem is not the enumeration of frequent subgraphs. Seki and Sese [18] introduced a problem to find the largest *connected* common pattern graph. However in the present paper, we focus on enumerating frequent *disconnected* graphs.

6 Concluding Remarks

In this paper, we introduced a novel problem called *ISS set enumeration problem*, which enumerates a set of large disconnected subgraphs in which all vertices share a large common itemset. The problem has wide application such as in side effect analysis for drug discovery and in viral-marketing effect investigations. However, it is difficult to find the graphs because of the difficulty of handling itemsets and a graph structure simultaneously. We designed a novel algorithm called *ROBIN* in order to solve this problem efficiently. Our demonstration with synthetic data showed that our algorithm is effective even in the case of a large and dense graph. Using our method, we found interesting graphs and itemsets from both a biological network and a citation network. From a biological network, we demonstrated the applicability in biological research and drug discovery. From a citation network, we found interesting patterns indicating successful collaborative works.

The problem of finding ISS sets is quite general and applicable to other itemset-associated graphs, and we are going to extend the applications to the others such as marketing in social networks and text analyses with Web links.

Acknowledgement

This work was partially supported by KAKENHI (Grant-in-Aid for Scientific Research) on Priority Areas "Systems Genomics" from the Ministry of Education, Culture, Sports, Science and Technology of Japan. We thank Dr. Tsuyoshi Kato for fruitful discussions.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proc. 20th Int. Conf. Very Large Data Bases, VLDB, pp. 487–499 (1994)
2. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: SIGMOD '00, pp. 1–12 (2000)
3. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. Data Min. Knowl. Discov. 1(3), 259–289 (1997)
4. Inokuchi, A., Washio, T., Motoda, H.: An Apriori-based algorithm for mining frequent substructures from graph data. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 13–23. Springer, Heidelberg (2000)
5. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: ICDM 2001, pp. 313–320 (2001)
6. Yan, X., Han, J.: gspan: Graph-based substructure pattern mining. In: ICDM '02, pp. 721 (2002)
7. Basu, S., Bilenko, M., Mooney, R.J.: A probabilistic framework for semi-supervised clustering. In: KDD '04, pp. 59–68 (2004)
8. Hashimoto, K., Takigawa, I., Shiga, M., Kanehisa, M., Mamitsuka, H.: Incorporating gene functions as priors in model-based clustering of microarray gene expression data. Bioinformatics 24(16), i167–i173 (2008)

9. Shiga, M., Takigawa, I., Mamitsuka, H.: A spectral clustering approach to optimally combining numerical vectors with a modular network. In: KDD '07, pp. 647–656 (2007)
10. Bayardo, R.: Efficiently mining long patterns from databases. In: SIGMOD '98, pp. 85–93 (1998)
11. Gasch, A.P., et al.: Genomic expression programs in the response of yeast cells to environmental changes. Mol. Biol. Cell 11(12), 4241–4257 (2000)
12. Knowledge Discovery Laboratory, University of Massachusetts Amherst: The Proximity DBLP database, http://kdl.cs.umass.edu/data/dblp/dblp-info.html
13. Huan, J., Wang, W., Prins, J., Yang, J.: Spin: mining maximal frequent subgraphs from graph databases. In: KDD '04, pp. 581–586 (2004)
14. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: Advances in knowledge discovery and data mining, pp. 307–328 (1996)
15. Zaki, M.J., Hsiao, C.J.: Efficient algorithms for mining closed itemsets and their lattice structure. IEEE TKDE 17(4), 462–478 (2005)
16. Ulitsky, I., Shamir, R.: Identification of functional modules using network topology and high throughput data. BMC Systems Biology 1 (2007)
17. Moser, F., Colak, R., Rafiey, A., Ester, M.: Mining cohesive patterns from graphs with feature vectors. In: SDM '09 (2009)
18. Seki, M., Sese, J.: Identification of active biological networks and common expression conditions. In: BIBE '08 (2008)

A Framework for SQL-Based Mining of Large Graphs on Relational Databases

Sriganesh Srihari[1], Shruti Chandrashekar[2], and Srinivasan Parthasarathy[3]

[1] School of Computing, National University of Singapore, Singapore 117590
[2] New Jersey Institute of Technology, Newark, NJ 07102
[3] The Ohio State University, Columbus, OH 43210
srigsri@comp.nus.edu.sg, sc297@njit.edu, srini@cse.ohio-state.edu

Abstract. We design and develop an SQL-based approach for querying and mining large graphs within a relational database management system (RDBMS). We propose a simple *lightweight framework* to integrate graph applications with the RDBMS through a tightly-coupled *network layer*, thereby leveraging efficient features of modern databases. Comparisons with straight-up main memory implementations of two kernels - breadth-first search and quasi clique detection - reveal that SQL implementations offer an attractive option in terms of productivity and performance.

Keywords: Graph mining, SQL-based approach, Relational databases.

1 Introduction

Over the past few years data generated from real-world processes have increasingly attracted the attention of researchers from all domains. A lot of effort has gone into analyzing this data from different perspectives to extract valuable information. In this respect, mining of *graph data* has always demanded its share of lime-light. This is primarily because graphs are ubiquituous and many *real world* scenarios are modeled as graphs. For example, the physical interactions between proteins in an organism are modeled as a protein interaction graph with proteins as vertices and their interactions as edges. Protein interaction graphs are a major resource for knowledge discovery: detecting protein complexes, predicting protein functions and reliabilities of interactions [10].

There are many efficient techniques developed for storing and manipulating graphs in main memory (RAM): for traversals, answering reachability queries, mining frequent patterns, etc. [5] However, as more and more graph data is accumulated, it is not feasible to store and manipulate entire graphs in main memory. Therefore, graphs are stored on disks and efficiently fetched into main memory in parts for manipulation [2]. The computational power of processors is increasing, while the speed gap between main and secondary (disk) memories is widening. Therefore, graphs are compressed and stored on disks so that they can be retrieved in parts with as little I/O reads as possible, and uncompressed quickly in main memory [1]. To summarize, these approaches used to handle graphs can be classified broadly into two categories: (a) efficient storage and manipulation of graphs in main memory; (b) efficient storage and indexing of graphs on disks and their retrieval into main memory (out-of-core approach).

As graph sizes continue to grow larger, it will be interesting to look at alternative approaches to mine large graphs (any data in general). The **SQL-based approach** for integrating mining with RDBMS (relational database management systems) was proposed long back (in 1998) for association rule mining [9], followed by k-way join variants for association rule mining in 2003 [6], Subdue-based substructure mining in 2004 [3], and frequent subgraphs mining in 2008 [4]. The SQL-based approach proposed storing data in relational databases and mining it using SQL queries. Even though this approach was considered novel and promising, the idea was mainly constrained to transactional datasets, and never became popular for mining graphs. One probable reason, we believe, was the complications (awkwardness) involved in "mapping" graphs onto the *relational framework* of a RDBMS. This involved expressing the whole problem (graph data, storage and manipulation) declaratively (using SQL).

In spite of the non-trivial nature of the SQL-based approach, it can be very useful and promising. The RDBMS displays data to the designers and programmers as relational structures, while internally it stores this data on disk blocks using efficient disk-based data structures (example, B+ trees). Hence, if we can reasonably "map" graph data structures and algorithms onto relational structures, then we can leverage all the services RDBMS can offer: handling dynamic updates, buffer management, indexing and retrieval, and parallelism. After all, more than two decades of research has gone into making database systems fast, scalable, robust, and concurrent. Secondly, in many instances, main memory and out-of-core implementations can get exceedingly non-trivial. However, the development and deployment time of SQL-based code can be significantly shorter because one can avoid denormalizing data and storing into flat files prior to data mining, and also writting code for indexing and retrieval [9].

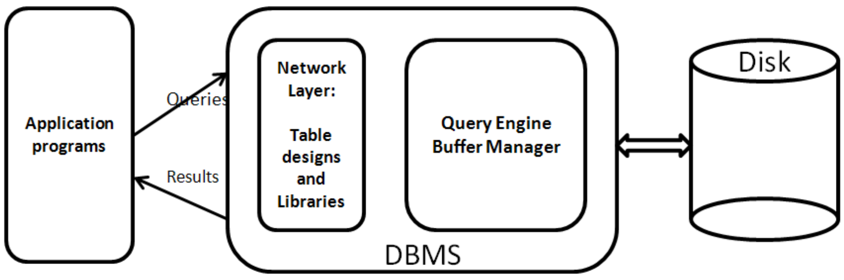


Fig. 1. Proposed framework for SQL-based mining of graphs on RDBMS

2 Our Proposed Framework

The main contribution of our work is to propose a **lightweight framework** for SQL-based mining of graphs on RDBMS. It is shown in Figure 1. This framework is designed for making effective use of the RDBMS features for efficient handling of large graphs. The **network layer** forms the most important component of the framework. Several graph mining applications can be "mapped" onto the RDBMS through the services offered by this layer.

2.1 The Network Layer

The network layer rests conceptually within the RDBMS (see Figure 1) and runs in the same address space as the RDBMS. It is implemented using procedural SQL (stored procedures using Oracle's PL/SQL [8]). The advantage of implementing this way is that the network layer is *tightly-coupled* to the RDBMS: it has direct access to all the services offered by the RDBMS. This layer provides the necessary *graph-abstraction* to abstract away all the complications involved in handling large graphs. It houses all the basic table designs and 'utilities'. Graph applications can either be implemented loosely-coupled or tightly-coupled to the RDBMS. For loosely-coupled applications, the network layer acts as a *translation layer* (For example, converting C or Java calls into SQL queries), while for tightly-coupled applications (written in procedural SQL), it provides ready-to-use libraries and utilities.

2.2 Efficient Storage of Graphs in the Network Layer

The basic schema design consists of storing all graphs $G = \{G_1, G_2, .., G_k\}$ in a hierarchical 'master-detail' fashion in the following tables: a graph table Graph($\underline{\text{GraphId}}$, NoOfVertices, NoOfEdges), a vertex table Vertex($\underline{\text{GraphId}}$, $\underline{\text{VertexId}}$), and a connectivity table AdjMatrix ($\underline{\text{GraphId}}$, $\underline{\text{Vertex1}}$, $\underline{\text{Vertex2}}$). For every graph $G_i = (V_i, E_i) \in G$, there is a record (tuple) in Graph, uniquely identified by the primary key $\{\text{GraphId}\} \leftarrow \{G_i\}$. For every vertex $v \in V_i$ of graph G_i, there is a record in Vertex, uniquely identified by primary key $\{\text{GraphId}, \text{VertexId}\} \leftarrow \{G_i, v\}$. The whole connectivity structure is then stored as records in AdjMatrix. For every edge $(u, v) \in E_i$, there is a record in AdjMatrix uniquely identified by the primary key $\{\text{GraphId}, \text{Vertex1}, \text{Vertex2}\} \leftarrow \{G_i, u, v\}$. Notice how *GraphId* is propagated as part of the primary key in all tables. The whole graph G_i can be uniquely queried from the tables using *GraphId*.

2.3 Implementing a Basic Utility within the Network Layer: BFS

We next describe how a basic utility like the breadth-first search (BFS) on a graph is efficiently implemented within the network layer.

The BFS algorithm on a graph $G_i = (V_i, E_i)$ and its SQL-based design are shown in Algorithm 1. We first store the graph G_i in the above-proposed tables. To simulate the FIFO queue used in BFS, we design a table Queue (Line: 1).

Algorithm 1. BFS(G, s)

```
1: Initialize queue $Q$; /* Create table: Queue(GraphId, VertexId, position) */
2: enqueue$(Q, s)$;
3: while $Q \neq$ empty do
4:    $v \leftarrow$ dequeue$(Q)$; /* Query: SELECT record with MIN position */
5:    for each unvisited neighbor $u$ of $v$ do
6:       enqueue$(Q, u)$; /* Insert into Queue. */
7:       mark $u$ as 'visited'; /* Update: 'visited' in Discovery. */
8:       assign a discovery number to $u$;
9:    end for
10:   if commitCnt $\geq$ commitFreq then
11:      COMMIT and reset commitCnt; /* Controlled COMMITs to restrict I/O.*/
12:   end if
13: end while
14: Output the vertices in discovery sequence;
```

For every vertex $v \in V_i$ that is enqueued, there is a record in Queue, uniquely identified by {GraphId, VertexId} $\leftarrow \{G_i, v\}$. The *position* attribute in Queue gives the position of v in the queue. The smaller the *position*, the earlier v will be dequeued. Additionally, for every vertex $v \in V_i$, there is a record in table Discovery, uniquely identified by the primary key {GraphId, VertexId} $\leftarrow \{G_i, v\}$. There are attributes *visited* and *discoveryNo* to keep track of whether v has been visited and its order in the visited sequence.

The BFS algorithm begins by inserting the source s into Queue (Line: 2). In each iteration, the vertex v with the minimum *position* is selected (Line: 4) from Queue. All unvisited neighbors u of v (Line: 5) are then selected from the join: AdjMatrix A $\bowtie_{A.Vertex1=v \wedge A.Vertex2=D.VertexId \wedge D.Visited=FALSE}$ Discovery D. These vertices are inserted into Queue (Line: 6) and updated as 'visited' in Discovery (Line: 7, 8). These iterations continue till Queue is empty.

2.4 Extending to Graph Mining: Quasi Clique Detection

Quasi cliques are very interesting structures from the point of view of graph mining. Very simply, a quasi clique in a graph is an 'almost' complete subgraph. Quasi cliques are used to model real-world communities in protein networks, social networks, scientific collaboration networks, etc. [10]

There are several ways to model quasi cliques; one way is by the notion of a *γ-quasi clique*. Given a graph $G = (V, E)$, a subgraph $Q = (V_Q, E_Q)$, $V_Q \subseteq V$ and $E_Q \subseteq E$, is called a quasi clique with clustering co-efficient $0 \leq \gamma \leq 1$ if, $|E_Q|$ is at least a γ-fraction of the total possible number of edges in a subgraph of the same size. This is given by: $|E_Q| \geq \gamma.\binom{|V_Q|}{2}$. Therefore, the number of edges missing in Q is given by: $\lambda \leq (1 - \gamma).\binom{|V_Q|}{2}$.

To study quasi clique detection on our framework, we chose the algorithm proposed in [10]. We only give the essense of the algorithm here so that the purpose of our work is served; for details see [10]. The inputs to the algorithm are graph $G = (V, E)$, and fixed *parameters* $k > 0$ and $\lambda \geq 0$. The algorithm performs a bounded recursive search to find a quasi clique $Q \subseteq V$ of size at most k with at most λ edges missing. The time complexity of the algorithm is $O(3^{k+\lambda}.f_{poly}(|V|))$. The recursive search procedure makes the algorithm highly memory-intensive with the amount of additional memory required per search-path being $O((k+\lambda).g_{poly}(|V|))$, which can be very large. This also reflects how non-trivial the memory management can be in such applications, especially when implemented in-memory or out-of-core.

2.5 The RCR Strategy in SQL-Based Approach

In order to implement the quasi clique algorithm using the SQL-based approach, we made use of the earlier proposed table designs. Additionally, we designed the following interesting strategy, which we call **replicate-cleanup-rebuild** (RCR). This strategy can be adopted to other recursive algorithms as well.

Algorithm 2. bool QCRecursive $(G, Q, V \setminus Q, k, \lambda)$: recursive call

```
1: $I = \{G, Q, V \setminus Q, k, \lambda\}$; /* Input $I$ from parent call.*/
2: $c$ = generateCallNo();
3: Working(GraphId, CallNo, Info) $\leftarrow \{G, c, I\}$; /* Replicate into Working. */
4: Pick an edge $(u, v)$;
5: $I' = \{G', Q' = Q \cup \{u\}, V \setminus Q', k' = k - 1, \lambda\}$; /* Include $u$ into solution.*/
6: if $Q'$ is the required solution then return TRUE along with $Q'$;
7: $r$ = QCRecursive$(G', Q', V \setminus Q', k', \lambda')$; /* Send new values to first child. */
8: if $r$ = TRUE then return TRUE along with the solution;
9: $I' = \emptyset$; /* Clean-up current values. */
10: $I' \leftarrow$ Working$(G, c)$; /* Rebuild from Working.*/
11: Repeat for subsequent children.
```

In this strategy, each call *replicates* (stores an additional copy) all the values received from its parent into a working table Working (see Algorithm 2). It makes its computations on the received values and passes the results to its child. When the child backtracks, instead of reverting back each computation, the current computed values are blindly *cleaned-up* (discarded), and the original values are *rebuilt* (queried) from Working. Subsequently, new computations are performed on these original values and sent to the next child. Also, when a child call backtracks, its records are permanently deleted from Working. The records stored for each call c are uniquely identified by the primary key {GraphId, CallNo} $\leftarrow \{G_i, c\}$ in Working. Considering $h_{poly}(|V|)$ number of records inserted per call, the total number of records in Working is $O((k+\lambda).h_{poly}(|V|))$.

Notice the intuition behind this strategy: to remove all the non-trivial memory management (local storage of values, and reverting back of computations from unsuccessful paths) within the calls and instead rely on the RDBMS for efficient mechanisms. It also illustrates how code development time can be significantly shorter using the SQL approach.

3 Empirical Evaluation

We compared SQL-based implementations against straight-up main memory implementations for: (a) breadth-first search (BFS) as a basic graph utility, and (b) quasi clique detection as a graph mining application. We implemented the main memory versions of the algorithms in C++ using the g++ 4.1.2 compiler on the GNU/Linux Debian distribution (2.6 kernel) installed on an Intel Xeon Dual Core 2.4GHz machine with 3GB RAM, 2.7GB swap partition and 250GB hard disk. Whenever the memory requirement was more than 3GB we relied on virtual memory. The procedural SQL versions were implemented in PL/SQL [8] on Oracle 10g on the same machine.

3.1 Evaluation of Breadth-First Search

We first compared the two implementations of BFS: (a) main memory (referred as BFSiMM) *versus* (b) procedural SQL (referred as BFSiSQL).

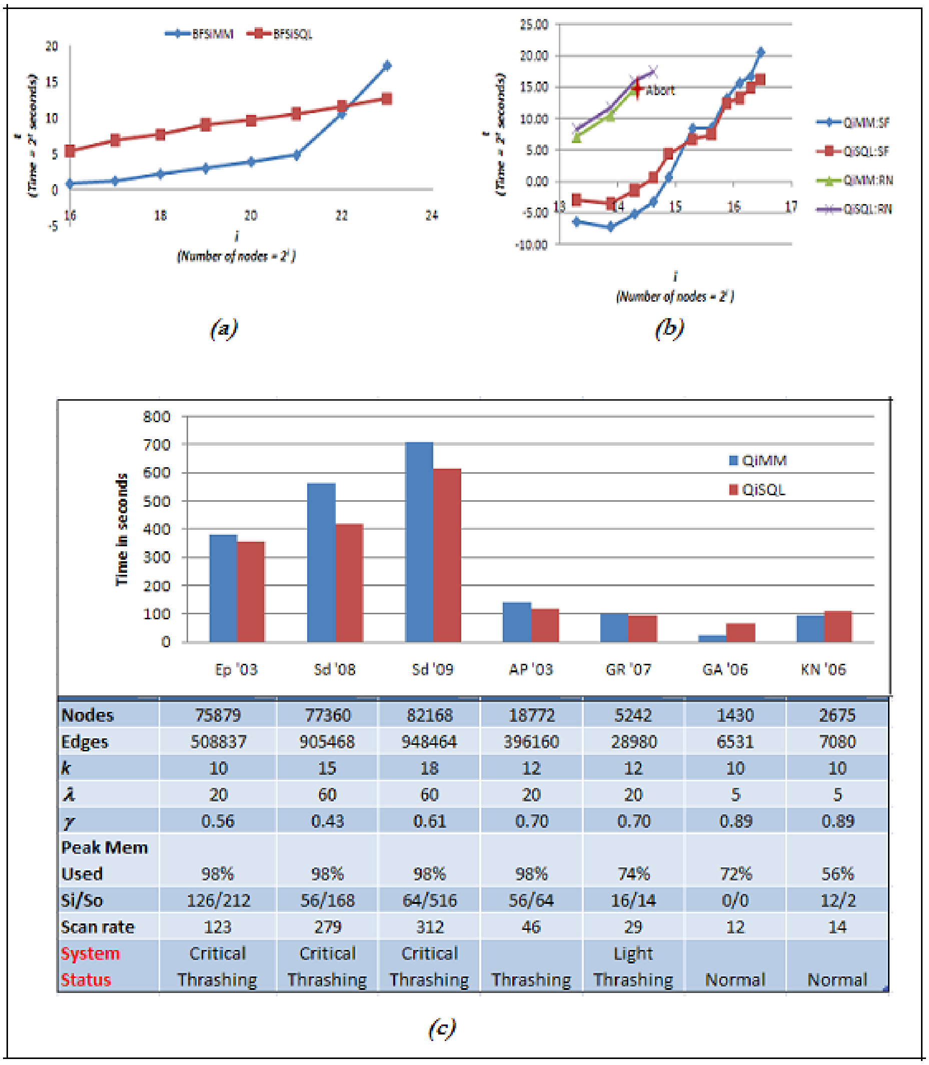
We generated random networks of n nodes and $m = 4n$ edges by replacement (that its, selecting m times nodes u and v such that $u \neq v$ and removing the edges between duplicated pairs). Figure 2(a) shows the comparison plots of runtimes (seconds) on networks for n between 2^{16} to 2^{23}. The figure shows that even though BFSiMM performed better than BFSiSQL for small networks, BFSiSQL outperformed BFSiMM for large networks.

3.2 Evaluation of Quasi Clique Detection

We next compared the two implementations of the quasi clique algorithm: (a) main memory (referred as QiMM) *versus* (b) procedural SQL (referred as QiSQL).

We generated scale-free networks with $n =$ 10K to 90K ($\sim 2^{13.28}$ to $\sim 2^{16.45}$), and random networks with $n =$ 10K to 40K ($\sim 2^{13.28}$ to $\sim 2^{15.28}$) vertices. We clustered them and stored co-clustered vertices on close-by disk blocks. Very small quasi cliques are easy to find and are not interesting, therefore we set $k = 25$ and $\lambda = 180$, giving $\gamma \geq \{\binom{k}{2} - \lambda\}/\binom{k}{2} = 0.4$. In each execution, 20 γ-quasi cliques were detected by iteratively deleting the current quasi clique and searching for the next one in the remaining network. Figure 2(b) shows the comparison of runtimes (in lg scale) for QiMM and QiSQL. It shows that even though QiMM performed better than QiSQL for small networks, QiSQL outperformed QiMM for large networks. For scale-free networks, this cross-over occurred around 60K ($\sim 2^{15.7}$) nodes. For random networks of size 25K ($\sim 2^{14.6}$), QiMM continuously aborted finding only 13 quasi cliques, while QiSQL found all 20 quasi cliques.

We next considered a variety of real-world networks obtained from [7]. These included social (Epinions: Ep'03, Slashdot: Sd'08 and Sd'09), scientific collaborations (Astro-physics: AP'03, General Relativity: GR'07) and protein interaction networks (Gavin: GA'06, Krogan: KN'06). Figure 2(c) shows the comparisons for fixed k and λ. It shows that QiSQL outperformed QiMM for all networks, except the small ones like PPI GA'06 and KN'06.



	Ep '03	Sd '08	Sd '09	AP '03	GR '07	GA '06	KN '06
Nodes	75879	77360	82168	18772	5242	1430	2675
Edges	508837	905468	948464	396160	28980	6531	7080
k	10	15	18	12	12	10	10
λ	20	60	60	20	20	5	5
γ	0.56	0.43	0.61	0.70	0.70	0.89	0.89
Peak Mem Used	98%	98%	98%	98%	74%	72%	56%
Si/So	126/212	56/168	64/516	56/64	16/14	0/0	12/2
Scan rate	123	279	312	46	29	12	14
System Status	Critical Thrashing	Critical Thrashing	Critical Thrashing	Thrashing	Light Thrashing	Normal	Normal

Fig. 2. (a) BFSiMM Vs BFSiSQL; (b) QiMM Vs QiSQL on scale-free and random networks; (c) QiMM Vs QiSQL on real-world networks

Analysis of deteriorating performance of QiMM: Even though the synthetic and real-world networks considered in the quasi clique experiments resided completely in main memory, QiMM displayed worse behavior compared to QiSQL for the larger networks. This was primarily because of the significant amount of additional memory required for recursive calls, which subjected QiMM to heavy *thrashing*. See Figure 2(c). *Snapshots* of memory usage (from *top* and *vmstat*) of the overall system when QiMM was executing showed 100% RAM and 100% CPU usage. The high swap-in (si) and swap-out (so) values (always zero while not thrashing) clearly indicated critical thrashing. The high scan indicated wastage of CPU cycles while waiting for the page handler to scan for free pages.

4 Conclusions and Future Work

In this work we have proposed a lightweight framework to extend the SQL-based approach to mine large graphs. We showed that this approach outperformed straight-up main memory implementations for BFS and quasi clique detection on large graph datasets. It will be interesting to realize our framework on grid technologies (like Oracle 10g/11g) for mining large graphs in a parallel distributed fashion.

Acknowledgements. We would like to thank Hon Wai Leong (NUS) and Anthony Tung (NUS) for the interesting discussions during this work, and the reviewers for their valuable comments. SP would like to acknowledge support from the following NSF grants – IIS-0917070, CCF-0702587, IIS-0347662. SS is supported under the NUS research scholarship.

References

1. Aggarwal, C., Yan, X., Yu, P.S.: GConnect: A connectivity index for massive disk-resident graphs. In: Very Large Databases (VLDB), vol. 2, pp. 862–873 (2009)
2. Chen, W., et al.: Scalable mining of large disk-based graph databases. In: ACM Knowledge Discovery and Data Mining (SIGKDD), pp. 316–325 (2004)
3. Chakravarthy, S., Beera, R., Balachandran, R.: DB-Subdue: Database approach to graph mining. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 341–350. Springer, Heidelberg (2004)
4. Chakravarthy, S., Pradhan, S.: DB-FSG: An SQL-based approach for frequent subgraph mining. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 684–692. Springer, Heidelberg (2008)
5. Jin, R., et al.: Efficiently answering reachability queries on very large directed graphs. In: ACM Management of Data (SIGMOD), pp. 595–608 (2008)
6. Mishra, P., Chakravarthy, S.: Performance evaluation and analysis of k-way join variants for association rule mining. In: James, A., Younas, M., Lings, B. (eds.) BNCOD 2003. LNCS, vol. 2712, pp. 95–114. Springer, Heidelberg (2003)
7. Network datasets, http://snap.stanford.edu/data/index.html
8. Oracle PL/SQL, http://www.oracle.com/technology/tech/pl_sql/index.html
9. Sarawagi, S., Thomas, S., Agarwal, R.: Integrating mining with relational database systems: Alternatives and implications. In: ACM Management of Data (SIGMOD), pp. 343–354 (1998)
10. Srihari, S., Ng, H.K., Ning, K., Leong, H.W.: Detecting hubs and quasi cliques in scale-free networks. In: IEEE Pattern Recognition (ICPR), pp. 1–4 (2008)

Fast Discovery of Reliable *k*-terminal Subgraphs

Melissa Kasari, Hannu Toivonen, and Petteri Hintsanen

Department of Computer Science and
Helsinki Institute for Information Technology HIIT
P.O. Box 68, FI–00014 University of Helsinki, Finland
{melissa.kasari,hannu.toivonen,petteri.hintsanen}@cs.helsinki.fi

Abstract. We present a novel and efficient algorithm for solving the most reliable subgraph problem with multiple query nodes on undirected random graphs. Reliable subgraphs are useful for summarizing connectivity between given query nodes. Formally, we are given a graph $G = (V, E)$, a set of query (or terminal) nodes $Q \subset V$, and a positive integer B. The objective is to find a subgraph $H \subset G$ containing Q, such that H has at most B edges, and the probability that H is connected is maximized. Previous algorithms for the problem are either computationally demanding, or restricted to only two query nodes. Our algorithm extends a previous algorithm to handle k query nodes, where $2 \leq k \leq |V|$. We demonstrate experimentally the usefulness of reliable k-terminal subgraphs, and the accuracy, efficiency and scalability of the proposed algorithm on real graphs derived from public biological databases.

1 Introduction

Graphs and networks are powerful means of representing information in various domains such as biology, sociology, and communications. However, large graphs are difficult to understand and use by humans. Given that the user is interested in some particular nodes and their connectivity, a large fraction of the original graph is often irrelevant. Subgraph extraction addresses this problem.

As an example application, consider *Biomine*, a biological graph consisting roughly of a million nodes and eight million edges [1]. One form of a query to Biomine is to specify a small number of query nodes, such as a gene and a disease, and extract a small subgraph that maximally connects the gene to the disease. A subgraph of few dozens of nodes typically already gives a good picture of the connectivity—not only the best paths, but a subgraph describing the network that connects the given nodes. At the same time, almost all of the millions of edges and nodes are irrelevant to how the gene is related to the disease.

In the *most reliable subgraph problem* [2], the user gives query nodes (also called terminals) and a budget, and the task is to extract a subgraph maximally relevant with respect to the given query nodes, but with a size within the given budget. The problem is defined for simple (Bernoulli) random graphs, where edge weights are interpreted as probabilities of the edges, and where a natural definition for "relevance" is network reliability (see Section 2).

In this paper, we propose a novel, efficient algorithm for extracting a reliable subgraph given an arbitrary number of query nodes. Previous work on the most reliable subgraph problem suffers either from a limitation to exactly two query nodes, or from computational complexity. Our work builds on a recent, efficient method for two query nodes, called *Path Covering* [3].

2 The Most Reliable *k*-terminal Subgraph Problem

We define the problem of finding the most reliable k-terminal subgraph, loosely following conventions and notations from previous work [4]. Let $G = (V, E)$ be an undirected graph where V is the set of nodes and E the set of edges. G is a Bernoulli random graph where each edge e has an associated probability p_e. The interpretation is that edge $e \in E$ exists with probability p_e, and conversely e does not exist, or is not true with probability $1 - p_e$. Given edge probabilities, the states of edges are mutually independent. Nodes are static.

Given a set $Q \subset V$ of nodes, or *terminals*, the *network reliability* $\mathrm{R}(G, Q)$ of G is defined as the probability that Q is connected, i.e., that any node in Q can be reached from any other node in Q [5]. In *the most reliable subgraph problem* we are looking for a subgraph $H \subset G$ connecting the terminals in Q, such that H has at most B edges and a maximal reliability with respect to the terminals, i.e., find $H^* = \arg\max_{H \subset G, ||H|| \leq B} \mathrm{R}(H, Q)$. Although the problem can be defined for directed graphs as well [2], we focus on undirected graphs in this paper. This problem, like reliability problems in general [6], is inherently difficult: efficient solutions are available only for restricted classes of graphs, but cases on general graphs are most likely intractable [2].

We now introduce some additional notation used in the later sections.

Given a graph G, $V(G)$ is the node set of G and $E(G)$ the edge set of G. Given a set of edges $S \subset E$, we say S *induces* a graph $G(S) = (V', S)$, where V' consists of the endpoints of the edges in S.

The union between two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is a new graph $H = G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. Other set operations for graphs are defined analogously. For notational convenience, we treat paths, trees and edges as (induced) graphs when there is no risk of confusion. This makes it notationally easy, e.g., to add a path P to a graph G by writing simply $G \cup P$ instead of $G \cup G(P)$, or to denote the edges of a tree T as $E(T)$ instead of $E(G(T))$.

Finally, a path with endpoints u and v is said to be a u–v-path.

Related Work. The most reliable subgraph problem was introduced recently [2], but algorithms were given only for the two-terminal case. We are aware of two previous solutions for the general case. One, proposed by Kroese et al., is based on the cross-entropy method [7]. De Raedt et al. give other solution to the general case in the setting of theory compression for ProbLog [8,9]. Unfortunately, these methods do not scale well to large databases, where input graphs may have hundreds or thousands of edges. Other closely related work includes connection subgraphs [10], center-piece subgraphs [11], and proximity graphs [12].

Recently, a novel Monte-Carlo sampling based algorithm Path Covering (PC) has been proposed for the two-terminal case [3]. The method proposed in this paper is based on the ideas of PC, so we briefly review its principles. The algorithm has two phases: a path sampling phase and a subgraph construction phase. In the path sampling phase, the goal is to identify a small set C of paths that have high probabilities and are relatively independent of each other. This is achieved by approximately maximizing the probability that at least one path $P \in C$ is true. We denote this probability by $\Pr(C) = \Pr(\bigvee_{P \in C} P)$.

In the subgraph construction phase, PC chooses a set of solution paths $S \subset C$ such that the probability $\Pr(S) = \Pr(\bigvee_{P \in S} P)$ is maximized and $||G(S)|| \leq B$, where $||G||$ denotes the number of edges in G. PC does not maximize $\mathrm{R}(G(S))$ directly, but works on its lower bound $\Pr(S)$ instead. Concisely put, PC generates S iteratively by choosing at each iteration the path P^* which gives the maximal per-edge increase to the (estimated) probability $\Pr(S)$, that is

$$P^* = \underset{P \in C \setminus S}{\arg\max} \frac{\Pr(S \cup P) - \Pr(S)}{|E(P) \setminus E(H)|}, \tag{1}$$

where $H = G(S)$ is the result subgraph being constructed. At each iteration, paths that become included into H are removed from C. To satisfy the budget constraint, paths $P \in C$ for which $||H|| + |E(P) \setminus E(H)| > B$ are also removed. The algorithm stops when $||H|| = B$ or $C \setminus S = \emptyset$, and returns the subgraph H.

3 Algorithms

We propose a novel, efficient algorithm for the problem of extracting a reliable k-terminal subgraph from an undirected graph. The proposed algorithm is a generalization of the Path Covering (PC) method [3] (see Section 2 for a brief overview) to more than two query nodes. The basic principles remain the same: the two phases, use of Monte Carlo simulations, as well as many subtle details. However, whereas PC uses paths connecting the two given query nodes as its building blocks (set C) in the subgraph construction phase, here we consider spanning trees connecting the k query nodes, with $2 \leq k \leq |V|$. Similarly, set S in the objective function (1) consists of spanning trees instead of paths as in PC.

In the first phase, the algorithm extracts a set of trees from the original graph G. Each of the trees connects the given k query nodes; by construction, they are spanning trees having the query nodes as leaves. In the second phase, these trees are used as building blocks to construct the result of the algorithm just like PC uses paths as its building blocks. We focus on the novel aspects of the proposed algorithm, the ones that allow solving the k-terminal problem. For brevity, we omit technical details shared with Path Covering and described in depth elsewhere [3]. We begin by presenting the general aspects of the new algorithm and then proceed to more detailed description.

Input and output data. The first phase of the algorithm (Algorithm 1) takes a random graph G and a set $Q \subset V$ of query nodes as its input. The algorithm outputs a set C of trees such that each tree connects all the query nodes. We call these trees *candidate trees*. C is used as an input in the second phase of the algorithm (Algorithm 2).

Producing candidate trees. At the first iteration, $(|Q|^2 - |Q|)/2$ new candidate trees are generated (Lines 2–3). Each tree connects one pair of query nodes and each query node pair is connected by one tree. Later, as the algorithm proceeds, new trees are added one by one; each of the later trees is also initially formed as a path between two query nodes. During the algorithm, individual trees are created and grown iteratively. In each iteration, either a branch is added to an existing *incomplete tree* (a tree that does not yet connect all query nodes) so that a new query node is connected to the tree, or a new initial tree is generated. At the end of the algorithm we output only *complete trees* (trees that connect all query nodes).

Edge sampling. The algorithm is stochastic. At each iteration it randomly decides, according to the probabilities p_e, which edges exist and which do not (Line 5). Only edges that are included in at least one candidate tree are decided. All other edges are considered to exist. The next step is to determine if any of the previous candidate trees exist in the current graph realization (Line 9). If one does not exist a new candidate tree is generated (Lines 14–15). If a previously discovered tree exists, the first such tree is taken into examination (Line 10). If the tree is complete, the algorithm proceeds directly to the next iteration. Otherwise the tree is extended (Lines 17–21) before continuing to the next iteration.

Tree construction. A new tree is formed by the best path connecting two query nodes (Line 15). A previously established incomplete tree is extended by connecting a new query node to it with the best path between some node in the tree and the new query node (Lines 18–20). The probabilities of all edges in the tree are set to 1 prior to the search of the best path (Line 17), while the probabilities of other edges remain the same. As a result the new branch is formed by the best path between the new query node and the tree. Edges that do not exist at the iteration are not used. All edge weights are set to their original values before proceeding to the next iteration (Line 21).

Choosing query nodes. When a new tree is formed, the algorithm decides randomly which two query nodes are included in the initial tree. Later on when an incomplete tree is extended, the algorithm again randomly selects the new query node to connect to the tree. This is to avoid unnecessary complexity: in our experiments this solution produced better results and shorter running times than selecting the node to be added based on its distance from the tree (results not shown).

Discovering strong trees. The collection C of candidate trees is organized as a queue, i.e., the oldest candidate trees are always considered first. This drives the algorithm to complete some trees first (the oldest ones) rather than extending them in random order and not necessarily up to a completion. On the other hand, the stochasticity of the algorithm favors strong trees: they are more likely to be true at any given iteration and thus more likely to be extended. The algorithm also has a tendency to avoid similar trees: when two (partial) trees are true at the same time, only the oldest one is potentially extended.

Algorithm 1. Sample trees

```
Input: Random graph $G$, set $Q \subset V$ of query nodes, number of trees to be generated
Output: Collection $C$ of trees connecting all query nodes
 1: $C \leftarrow \emptyset$
 2: for each node pair $v_i$, $v_j \in Q$, $v_i \neq v_j$ do
 3:     Add the best $v_i$–$v_j$-path from $G$ to $C$
 4: repeat
 5:     Decide each $e \in E(C)$ by flipping a coin biased by the probability of $e$
 6:     Let $E_S$ contain the successful edges and $E_F$ contain the failed edges
 7:     $T_S \leftarrow \emptyset$
 8:     for $i = 1$ to $|C|$ do
 9:         if $E(C_i) \subset E_S$ then
10:             if $Q \subset V(C_i)$ then
11:                 continue at line 5
12:             $T_S \leftarrow C_i$
13:             continue at line 17
14:     Randomly select $u$ and $v \in Q$, $u \neq v$
15:     Add the best $u$–$v$-path from $G - E_F$ to $C$
16:     continue at line 5
17:     Set the probability of $e$ to 1 for all $e \in E(T_S)$
18:     Randomly select $u \in Q \cap V(T_S)$ and $v \in Q \setminus V(T_S)$
19:     $P_S \leftarrow$ the best $u$–$v$-path from $G - E_F$
20:     Add all $e \in E(P_S)$ to $E(T_S)$ and all $v \in V(P_S)$ to $V(T_S)$
21:     Reset edge weights for all $e \in E(T_S)$
22: until the stopping condition is satisfied
23: return $C$ (after removing all incomplete trees)
```

Stopping condition. We use the number $|C|$ of complete trees generated as the stopping condition for the first phase. The number of iterations would be another alternative (see Section 4). Using the number of trees seems a better choice than using the number of iterations, since the minimum number of iterations needed to produce a single complete tree increases when the number of query nodes increase.

4 Experiments

We have implemented and experimentally evaluated the proposed algorithm on random graphs derived from public biological databases. In this section, we examine the usefulness of k-terminal subgraphs: do they maintain the k-terminal reliability of input graphs, and what is the amount of random variance in the results? We demonstrate the efficiency and scalability of our algorithm for large input graphs. Finding a suitable stopping criterion for Algorithm 1 is difficult; we also address this issue. Finally, we compare the algorithm against a baseline method based on enumerating the best paths between the query nodes.

Algorithm 2. Select trees

```
Input: Random graph $G$, collection $C$ of trees, budget $B \in \mathbb{N}$
Output: Subgraph $H \subset G$ with at most $B$ edges
1: $S \leftarrow \emptyset$
2: while $||G(S)|| \leq B$ and $C \setminus S \neq \emptyset$ do
3:     Find $T = \arg\max_{T \in C \setminus S} \frac{\Pr(S \cup T) - \Pr(S)}{|E(T) \setminus E(G(S))|}$
4:     $S \leftarrow S \cup T$
5:     for all $T \in C$ do {remove useless trees}
6:         if $T \subset G(S)$ or $||G(S)|| + |E(T) \setminus E(G(S))| > B$ then
7:             $C \leftarrow C \setminus \{T\}$
8: return $H = G(S)$
```

4.1 Test Set-Up

Test data and query nodes. We use *Biomine* database [1] as our data source. Biomine is a large index of various interlinked public biological databases, such as EntrezGene, UniProt, InterPro, GO, and STRING. Biomine offers a uniform view to these databases by representing their contents as a large, heterogeneous biological graph. Nodes in this graph represent biological entities (records) in the original databases, and edges represent their annotated relationships. Edges have weights, interpreted as probabilities [1]. We evaluated the proposed method using six source graphs of varying sizes (Table 1) and a set of up to ten query nodes. They were obtained as follows.

Table 1. Sizes of source graphs used in the experiments

Name of $G = (V, E)$	**400**	**500**	**700**	**1000**	**2000**	**5000**
Number of edges, $\lvert E\rvert = \lVert G\rVert$	395	499	717	1046	2019	4998
Number of nodes, $\lvert V\rvert$	153	189	260	336	579	1536
Reliability of G with $\lvert Q\rvert = 4$	0.46	0.46	0.50	0.56	0.59	0.60

First, the largest subgraph, consisting of approximately 5000 edges and 1500 nodes, was retrieved from the Biomine database using *Crawler*, a subgraph retrieval component proprietary to Biomine. For this initial retrieval, we used eight randomly selected query nodes.

Second, a set of ten query nodes to be used in the experiments was defined by randomly picking nodes from the subgraph of 5000 edges. The query node identifiers are EntrezGene:348, EntrezGene:29244, EntrezGene:6376, EntrezGene:4137, UniProt:P51149, UniProt:Q91ZX7, EntrezGene:14810, UniProt: P49769, EntrezGene:11810, and UniProt:P98156.

Third, smaller subgraphs were retrieved with Crawler by a sequence of subgraph retrievals, always extracting the next smaller subgraph from the previous subgraph, using the ten query nodes given above.

We also used two additional source graphs when evaluating the scalability of the algorithm to large source graphs. The smaller one consisted of 51,448 edges and 15,862 nodes. The size of the larger graph was 130,761 edges and 66,990 nodes. The smaller graph was a subgraph of the larger one and all other source graphs used in the experiments were subgraphs of both.

Biomine Crawler. The subgraph retrieval component of the Biomine system, "Crawler", was used to extract the source graphs, and it will also be used below in a comparative experiment to assess the effectiveness of the proposed algorithm. Crawler is currently undocumented. Given a source graph, a set of query nodes, and a node budget, it works roughly as follows. It first finds a large set of best paths between all pairs of query nodes. It then picks those paths sequentially between the node pairs, until the subgraph induced by the chosen paths reaches the specified number of nodes. The method is somewhat similar to the BPI algorithm [4], but works for multiple query nodes. Even though the Crawler works with random graphs, it does not try to optimize the k-terminal reliability.

Stopping condition. We used the number of complete candidate trees generated as the stopping condition for Algorithm 1. Another obvious alternative would be the number of iterations. Neither condition is perfect: for instance, the number of query nodes has a strong effect on the number of trees needed to find a good subgraph. On the other hand, the number of query nodes has also a strong effect on the number of iterations needed to produce a sufficient amount of trees. A single fixed number of candidate paths is a suitable stopping condition for the two-terminal case [3] but it is problematic in the k-terminal case where the building blocks are trees consisting of multiple branches. For the current experiments, we believe a fixed number of candidate trees gives a fair impression of the performance of the method.

Parameters. The experiments have been performed using the following parameter values; the default values we have used throughout the experiments, unless otherwise indicated, are given in boldface.

– Size of source graph G (Table 1): $||G||$ = 400, **500**, 700, 1000, 2000, 5000
– Number of query nodes: $|Q|$ = 1, 2, 3, **4**, ..., 10
– Size of extracted subgraph: $||H||$ = 10, 20, 30, ..., **60**, ..., 100, 150, 200
– Number of complete trees (stopping condition of Algorithm 1): $|C|$ = 10, 20, 30, ..., **100**, 150, 200

To control random variation, the values reported below are averages over 10 independent runs.

4.2 Results

Let us first take a look at how well the method manages to extract a reliable subgraph (Fig. 1). For three to four query nodes ($|Q|$), a subgraph of only 20–30 edges manages to capture 80% of the reliability of the source graph of 500 edges. For a large number of query nodes, the problem seems much more challenging. It seems obvious, that larger subgraphs are needed for a larger number of query nodes, if the reliability should be preserved.

The number $|C|$ of candidate trees produced in the first phase of the algorithm has an effect on the reliability of the extracted subgraph, but we discovered that sampling a relatively small number of trees is enough to produce good subgraphs (approximately 50 trees for four query nodes; results not shown). An experimental analysis of the running time indicates that the method scales linearly with respect to the number of candidate trees generated.

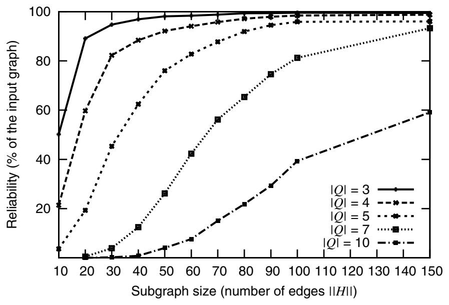


Fig. 1. Relative reliability of a subgraph as a function of its size, for various numbers of query nodes

The scalability of the new algorithm to large source graphs (Fig. 2, left) is clearly superior to previous methods (see Section 1). Source graphs of thousands of edges are handled within a second or two. Scalability is close to linear, which is expected: the running time of the algorithm is dominated by Monte-Carlo simulation, whose complexity grows linearly with respect to the input graph size and the number of iterations. Running times for the two additional large source graphs (51,448 edges and 130,761 edges) are not shown in the figure, but the average running times over ten independent runs are approximately 16 (standard deviation 0.57) and 53 seconds (standard deviation 2.8), respectively. Limiting the length of tree branches might shorten running times in some cases.

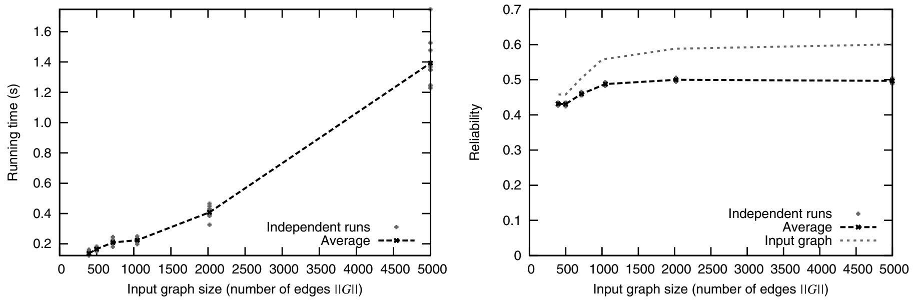


Fig. 2. Running time and reliability as functions of the size of the input graph

The right panel of Fig. 2 indicates the original reliability of the growing source graph, as well as the reliability of the extracted subgraph (of a fixed size of 60 edges). The relative difference in reliability is less than 20% in all cases, emphasizing the ability of the algorithm to preserve strong connectivity between the query nodes. These results suggest that the algorithm is competitive for interactive visualization.

Finally, we compare the new algorithm to the Biomine Crawler (Fig. 3), as it is the only available method for comparison on this scale. The comparison is not completely fair, as the Crawler does not aim to optimize the k-terminal reliability, but the general goal of extracting a subgraph connecting the query nodes is the same. In the experiments, the proposed algorithm produces significantly more reliable subgraphs, especially when the extracted subgraphs are small. Both methods converge towards the reliability of the source graph. However, where the new method reaches 80% of the original reliability with only 30 edges, the Crawler needs 60 edges for the same.

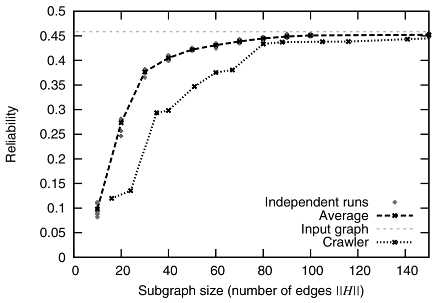


Fig. 3. Comparison with Biomine Crawler with 4 query nodes

5 Conclusions

We gave an efficient algorithm for solving the most reliable subgraph extraction problem for more than two query nodes. This is a significant improvement over state-of-the-art that could efficiently only handle exactly two query nodes.

Experimental results with real biological data indicate that the problem and the proposed method have some very useful properties. First, reliable k-terminal subgraphs of fairly small sizes seem to capture essential connectivity well. Second, the proposed method extracts a reliable subgraph in a matter of seconds, even from a source graph of thousands of edges; the time complexity seems to be linear with respect to the size of the original graph.

There are many possible variants of the approaches described in this paper that could be explored to find better solutions. For instance, how to choose which partial tree to expand and how to expand it, or how to efficiently use partial trees also in the second phase? Another interesting approach could be using (approximated) Steiner trees as spanning trees.

Future experiments include systematic tests to find out robust sets of parameters that perform reliably over wide range of input graphs and query nodes, and more extensive comparisons with related methods. Possible extension of the proposed algorithm for directed variant of the problem is an open question.

Acknowledgements. We would like to thank the Biomine team and especially Lauri Eronen for providing the Biomine Crawler. This work has been supported by the Algorithmic Data Analysis (Algodan) Centre of Excellence of the Academy of Finland (Grant 118653).

References

1. Sevon, P., Eronen, L., Hintsanen, P., Kulovesi, K., Toivonen, H.: Link discovery in graphs derived from biological databases. In: Leser, U., Naumann, F., Eckman, B. (eds.) DILS 2006. LNCS (LNBI), vol. 4075, pp. 35–49. Springer, Heidelberg (2006)
2. Hintsanen, P.: The most reliable subgraph problem. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 471–478. Springer, Heidelberg (2007)
3. Hintsanen, P., Toivonen, H., Sevon, P.: Link discovery by reliable subgraphs (submitted 2010)
4. Hintsanen, P., Toivonen, H.: Finding reliable subgraphs from large probabilistic graphs. Data Mining and Knowledge Discovery 17, 3–23 (2008)
5. Colbourn, C.J.: The Combinatorics of Network Reliability. Oxford University Press, Oxford (1987)
6. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM Journal on Computing 8, 410–421 (1979)
7. Kroese, D.P., Hui, K.P., Nariai, S.: Network reliability optimization via the cross-entropy method. IEEE Transactions on Reliability 56, 275–287 (2007)
8. Raedt, L.D., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic prolog and its application in link discovery. In: Proceedings of the 12th International Joint Conference on Artificial Intelligence, pp. 2468–2473 (2007)
9. Raedt, L.D., Kersting, K., Kimmig, A., Revoredo, K., Toivonen, H.: Compressing probabilistic Prolog programs. Machine Learning 70, 151–168 (2008)
10. Faloutsos, C., McCurley, K.S., Tomkins, A.: Fast discovery of connection subgraphs. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 118–127 (2004)
11. Tong, H., Faloutsos, C.: Center-piece subgraphs: problem definition and fast solutions. In: Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 404–413 (2006)
12. Koren, Y., North, S.C., Volinsky, C.: Measuring and extracting proximity graphs in networks. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 245–255 (2006)

GTRACE2: Improving Performance Using Labeled Union Graphs

Akihiro Inokuchi[1,2] and Takashi Washio[1]

[1] The Institute of Scientific and Industrial Research, Osaka University
Mihogaoka 8-1, Ibaraki, Osaka 567-0047, Japan
[2] PRESTO, Japan Science and Technology Agency
{inokuchi,washio}@ar.sanken.osaka-u.ac.jp

Abstract. The mining of a complete set of frequent subgraphs from labeled graph data has been studied extensively. Recently, much attention has been given to frequent pattern mining from graph sequences. In this paper, we propose a method to improve GTRACE which mines frequent patterns called FTSs (Frequent Transformation Subsequences) from graph sequences. Our performance study shows that the proposed method is efficient and scalable for mining both long and large graph sequence patterns, and is some orders of magnitude faster than the conventional method.

Keywords: Frequent Pattern, Graph Sequence, Labeled Union Graph.

1 Introduction

Studies on data mining have established many approaches for finding characteristic patterns from various structured data. Graph Mining [5,12,9], which efficiently mines all subgraphs appearing more frequently than a given threshold from a set of graphs, focuses on the topological relations between vertices in the graphs. Although the major methods for Graph Mining are quite efficient in practice, they require much computation time to mine complex frequent subgraphs due to the NP-completeness of subgraph isomorphism matching [4]. Accordingly, these conventional methods are not suitable for complex graphs such as graph sequences.

However, graph sequences can be used to model objects for many real world applications. For example, a human network can be represented as a graph where each human and each relationship between two humans correspond to a vertex and an edge, respectively. If a human joins (or leaves) the community in the human network, the numbers of vertices and edges in the graph increase (or decrease). Similarly, a gene network consisting of genes and their interactions produces a graph sequence in the course of their evolutionary history by acquiring new genes, deleting genes, and mutating genes.

Recently, much attention has been given to frequent pattern mining from graph sequences [6,2,1,7]. Figure 1 (a) shows an example of a graph sequence consisting of 4 steps where each contains vertices denoted by 5 unique IDs.

In [6], we proposed a new method, called GTRACE (Graph TRAnsformation sequenCE mining), for mining frequent patterns as shown in Fig. 1 (b) from graph sequences under the assumption that the change in each graph is gradual, and applied it to graph sequences generated from the Enron dataset. Although GTRACE is tractable for the Enron graph sequences containing about 7 steps and 100 unique IDs, it is intractable for graph sequences containing more steps and unique IDs than those in the Enron graph sequences.

In this paper, we propose a method to improve the efficiency of GTRACE mining frequent patterns called FTSs (Frequent Transformation Subsequences) from graph sequences. Our performance study shows that the proposed method is efficient and scalable for mining both long and large graph sequence patterns, and is some orders of magnitude faster than GTRACE. Although this paper focuses on undirected graphs where only the vertices have labels, the proposed method is applicable to both directed graph and undirected graphs where the edges also have labels without loss of generality.

2 Representation of Graph Sequences

In this section, we briefly review a compilation used to compactly represent graph sequences in GTRACE. Figure 1 (a) shows an example of a graph sequence. The graph $g^{(j)}$ is the j-th labeled graph in the sequence. The problem we address in this paper is how to mine patterns that appear more frequently than a given threshold from a set of graph sequences. In [6], we proposed transformation rules to represent graph sequences compactly under the assumption that "the change over adjacent graphs is gradual". In other word, only a small part of the graph changes between two successive graphs $g^{(j)}$ and $g^{(j+1)}$ in a graph sequence, while the other parts remain unchanged. In the aforementioned human networks and the gene networks, these assumptions certainly hold, since most of the changes of the vertices are progressive over successive steps. The direct representation of a graph sequence is not compact, because many parts of a graph remain unchanged over several steps and are therefore redundant in the representation. On the other hand, a graph sequence is compactly represented by introducing a representation of graph transformation based on rules of insertion, deletion, and relabeling of vertices and edges under the assumption of gradual changes.

A labeled graph g is represented as $g = (V, E, L, f)$, where $V = \{v_1, \cdots, v_z\}$ is a set of vertices, $E = \{(v, v') \mid (v, v') \in V \times V\}$ is a set of edges, and L is a set of labels such that $f : V \to L$. $V(g)$, $E(g)$, and $L(g)$ are sets of vertices, edges and labels of g, respectively. A graph sequence is represented as $d = \langle g^{(1)}\ g^{(2)} \cdots g^{(n)} \rangle$, where the superscript integer of each g is the ordered step in

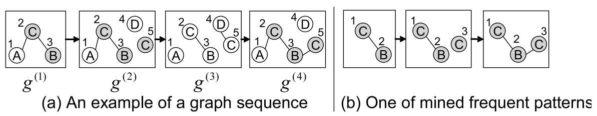

$g^{(1)}$
$g^{(2)}$
$g^{(3)}$
$g^{(4)}$
(a) An example of a graph sequence
(b) One of mined frequent patterns

Fig. 1. Examples of a graph sequence and a mined frequent pattern

the graph sequence. We assume that each vertex v is mutually distinct from the others in any $g^{(j)}$ and keeps a unique ID $id(v)$ in d. We define the set of unique IDs to be $ID_V(d) = \{id(v) \mid v \in V(g^{(j)}), g^{(j)} \in d\}$ and the set of pairs of unique IDs to be $ID_E(d) = \{(id(v), id(v')) \mid (v, v') \in E(g^{(j)}), g^{(j)} \in d\}$.

Example 1. In the human network mentioned in Section 1, each person has a unique ID, and his/her gender is an example of a vertex label.

To compactly represent a graph sequence, we focus on the differences between two successive graphs $g^{(j)}$ and $g^{(j+1)}$ in the sequence.

Definition 1. *Given a graph sequence $d = \langle g^{(1)} \cdots g^{(n)} \rangle$, each graph $g^{(j)}$ in d is called an "interstate". Moreover, The differences between the graphs $g^{(j)}$ and $g^{(j+1)}$ are interpolated by a virtual sequence $\langle g^{(j,1)} g^{(j,2)} \cdots g^{(j,m_j)} \rangle$, where $g^{(j,1)} = g^{(j)}$ and $g^{(j,m_j)} = g^{(j+1)}$. Each graph $g^{(j,k)}$ is called an "intrastate". The graph sequence d is represented by the interpolations as $d = \langle s^{(1)} s^{(2)} \cdots s^{(n-1)} \rangle$.* ■

The order of interstates represents the order of graphs in a sequence. On the other hand, the order of intrastates is the order of graphs in the artificial interpolation, and there can be various interpolations between the graphs $g^{(j)}$ and $g^{(j+1)}$. We limit the interpolations to be compact and unambiguous by choosing one with the shortest length in terms of graph edit distance.

Definition 2. *Let a transformation of a graph by insertion, deletion or relabeling of a vertex or an edge be a unit, and let each unit have edit distance 1. An "intrastate sequence" $s^{(j)} = \langle g^{(j,1)} g^{(j,2)} \cdots g^{(j,m_j)} \rangle$ is defined as the interpolation in which the edit distance between any two successive intrastates is 1, and in which the edit distance between any two intrastates is minimum.* ■

The transformation is represented by the following "transformation rule (TR)".

Definition 3. *A transformation rule (TR) which transforms $g^{(j,k)}$ to $g^{(j,k+1)}$ is represented by $tr^{(j,k)}_{[o_{jk}, l_{jk}]}$, where*

- *tr is a transformation type which is either insertion, deletion, or relabeling of a vertex or an edge,*
- *o_{jk} is an element in $ID_V(d) \cup ID_E(d)$ to be transformed, and*
- *$l_{jk} \in L$ is a label to be assigned to the vertex by the transformation.* ■

For the sake of simplicity, we denote the transformation rule by omitting the subscripts of o_{jk} and l_{jk} except in the case of ambiguity. We introduced five TRs defined in Table 1. In summary, we give the following definition of a transformation sequence.

Definition 4. *An intrastate sequence $s^{(j)} = \langle g^{(j,1)} \cdots g^{(j,m_j)} \rangle$ is represented by $seq(s^{(j)}) = \langle tr^{(j,1)}_{[o,l]} \cdots tr^{(j,m_j-1)}_{[o,l]} \rangle$. This is called an "intrastate transformation sequence". Moreover, a graph sequence $d = \langle g^{(1)} \cdots g^{(n)} \rangle$ is represented by an "interstate transformation sequence" $seq(d) = \langle seq(s^{(1)}) \cdots seq(s^{(n-1)}) \rangle$.* ■

Table 1. Transformation rules (TRs) to represent graph sequence data

Vertex Insertion $vi_{[u,l]}^{(j,k)}$	Insert a vertex with label l and unique ID u into $g^{(j,k)}$ to transform to $g^{(j,k+1)}$.
Vertex Deletion $vd_{[u,\bullet]}^{(j,k)}$	Delete an isolated vertex with unique ID u in $g^{(j,k)}$ to transform to $g^{(j,k+1)}$.
Vertex Relabeling $vr_{[u,l]}^{(j,k)}$	Relabel a label of a vertex with unique ID u in $g^{(j,k)}$ to be l to transform to $g^{(j,k+1)}$.
Edge Insertion $ei_{[(u_1,u_2),\bullet]}^{(j,k)}$	Insert an edge between 2 vertices with unique IDs u_1 and u_2 into $g^{(j,k)}$ to transform to $g^{(j,k+1)}$.
Edge Deletion $ed_{[(u_1,u_2),\bullet]}^{(j,k)}$	Delete an edge between 2 vertices with unique IDs u_1 and u_2 in $g^{(j,k)}$ to transform to $g^{(j,k+1)}$.

Arguments l of the transformations of vertex deletion vd, edge insertion ei, and edge deletion ed are dummy and represented by '$\bullet$'.

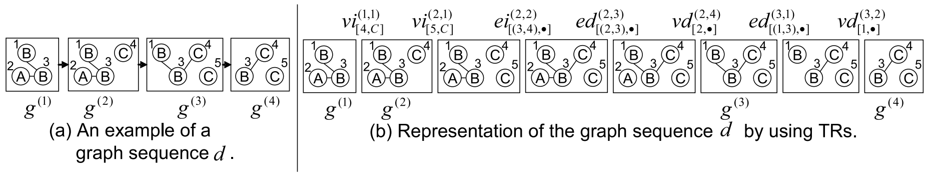

$vi_{[4,C]}^{(1,1)}$
$vi_{[5,C]}^{(2,1)}$
$ei_{[(3,4),\bullet]}^{(2,2)}$
$ed_{[(2,3),\bullet]}^{(2,3)}$
$vd_{[2,\bullet]}^{(2,4)}$
$ed_{[(1,3),\bullet]}^{(3,1)}$
$vd_{[1,\bullet]}^{(3,2)}$
$g^{(1)}$
$g^{(2)}$
$g^{(3)}$
$g^{(4)}$
(a) An example of a graph sequence d.
(b) Representation of the graph sequence d by using TRs.

Fig. 2. A graph sequence and its TRs

The notation of the intrastate transformation sequence is far more compact than the original graph based representation, since only differences between two successive intrastates appear in the sequence. In addition, computing a sequence of TRs based on differences between two graphs is solvable in linear time, because all vertices have unique IDs.

Example 2. In Fig. 2 (a), a graph sequence is represented by a sequence of insertions and deletions of vertices and edges as shown in Fig. 2 (b). The sequence is compiled into $\langle vi_{[4,C]}^{(1,1)} vi_{[5,C]}^{(2,1)} ei_{[(3,4),\bullet]}^{(2,2)} ed_{[(2,3),\bullet]}^{(2,3)} vd_{[2,\bullet]}^{(2,4)} ed_{[(1,3),\bullet]}^{(3,1)} vd_{[1,\bullet]}^{(3,2)} \rangle$.

3 Mining Frequent Transformation Subsequences

In this section, we briefly review how GTRACE mines frequent transformation subsequences (FTSs) from a given set of graph sequences. To mine FTSs from a set of compiled graph sequences, we define an inclusion relation between transformation sequences. When a transformation sequence $seq(d)$ includes another transformation sequence $seq(d')$, it is denoted by $seq(d') \sqsubseteq seq(d)$ whose detail definition is provided in [6].

As mentioned in [6], to mine FTSs consisting of mutually relevant vertices only, we define the relevancy between unique IDs of vertices and edges as follows.

Definition 5. *Unique IDs in $d = \langle g^{(1)} \cdots g^{(n)} \rangle$ are relevant one another, and d is called a "relevant graph sequence", if the union graph $g_u(d)$ of d is a connected graph. We define the union graph of d to be $g_u(d) = (V_u, E_u)$ where*

$$V_u = \{id(v) \mid v \in V(g^{(j)}), g^{(j)} \in d\}, and \tag{1}$$
$$E_u = \{(id(v), id(v')) \mid (v, v') \in E(g^{(j)}), g^{(j)} \in d\}. \tag{2}$$

■

This union graph of the transformation sequence $seq(d)$ is also defined similar to Definition 5.

Given a set of data $DB = \{\langle id, d\rangle \mid d = \langle g^{(1)} \cdots g^{(n)}\rangle\}$, the support value $\sigma(seq(d'))$ of a transformation subsequence $seq(d')$ is defined to be

$$\sigma(seq(d')) = |\{id \mid \langle id, d\rangle \in DB, seq(d') \sqsubseteq seq(d)\}|.$$

We call a transformation subsequence whose support value is greater than or equal to a minimum support threshold σ' a "frequent transformation subsequence (FTS)". The anti-monotonicity of this support value holds. That is, if $seq(d'_1) \sqsubset seq(d'_2)$ then $\sigma(seq(d'_1)) \geq \sigma(seq(d'_2))$. Using these settings, we state our mining problem as follows.

Problem 1. Given a dataset $DB = \{\langle id, d\rangle \mid d = \langle g^{(1)} g^{(2)} \cdots g^{(n)}\rangle\}$ and a minimum support threshold σ' as the input, enumerate all relevant FTSs (rFTSs).

To enumerate all rFTSs efficiently, GTRACE first generates a union graph for each graph sequence in DB based on the definition of a union graph. Subsequently, all connected frequent subgraphs in these union graphs are enumerated by using the conventional Graph Mining algorithm. At each time the algorithm outputs a connected frequent subgraph, an altered version of PrefixSpan [10] is called to mine rFTSs from transformation subsequences generated by the following projection.

Definition 6. *Given a graph sequence $\langle id, d\rangle \in DB$ and a connected graph g, we define a function "$proj_1$" to project $seq(d)$ to its subsequences.*

$$proj_1(\langle id, d\rangle, g) = \{\langle id', seq(d')\rangle \mid id = id', seq(d') \sqsubseteq seq(d) \text{ s.t. } g_u(d') = g\}. \quad ■$$

A data ID id' is attached to each transformation subsequence produced by the projection to calculate the exact support value of each rFTS, since multiple transformation subsequences are produced from a graph sequence $\langle id, d\rangle$ in the projection. Since the union graph of an rFTS is also frequent in the union graphs of all $\langle id, d\rangle \in DB$, we can enumerate all rFTSs from the projected transformation subsequences if all connected frequent subgraphs among the union graphs of all $\langle id, d\rangle$ are given.

Example 3. Given the graph sequence d in Fig. 3 (a), $seq(d)$ is represented by $seq(d) = \langle \underline{vi^{(1,1)}_{[3,C]} ei^{(1,2)}_{[(1,3),\bullet]} ei^{(1,3)}_{[(2,3),\bullet]}} vi^{(2,1)}_{[4,A]} ei^{(2,2)}_{[(1,4),\bullet]} \underline{ed^{(2,3)}_{[(1,3),\bullet]}} \rangle$, and its union graph $g_u(d)$ is depicted in Fig. 3 (b). Given a graph g which is a subgraph of $g_u(d)$ as shown in Fig. 3 (d), one of transformation sequences in $proj_1(\langle id, seq(d)\rangle, g)$ is $\langle id, seq(d')\rangle = \langle id, \langle vi^{(1,1)}_{[3,C]} ei^{(1,2)}_{[(1,3),\bullet]} ei^{(1,3)}_{[(2,3),\bullet]} ed^{(2,1)}_{[(1,3),\bullet]} \rangle\rangle$ as depicted in Fig. 3 (c), where this subsequence matches with the underlined rules in $seq(d)$.

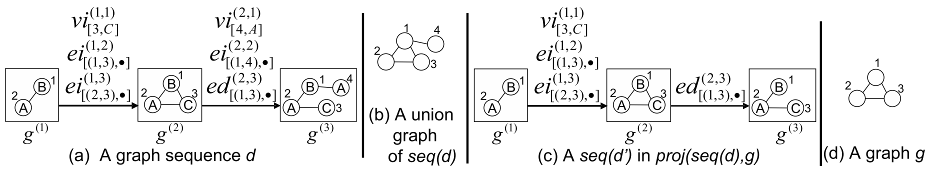

$vi^{(1,1)}_{[3,C]}$
$ei^{(1,2)}_{[(1,3),\bullet]}$
$ei^{(1,3)}_{[(2,3),\bullet]}$
$vi^{(2,1)}_{[4,A]}$
$ei^{(2,2)}_{[(1,4),\bullet]}$
$ed^{(2,3)}_{[(1,3),\bullet]}$
$g^{(1)}$
$g^{(2)}$
$g^{(3)}$
(a) A graph sequence *d*
(b) A union graph of *seq(d)*
(c) A *seq(d')* in *proj(seq(d),g)*
(d) A graph *g*

Fig. 3. An example of the projection

```
1) GTRACE($DB$, $\sigma'$)
2)     $G_u = \{g_u(d) \mid \langle id, d\rangle \in DB\}$
3)     for $g$ =FrequentSubgraphMiner($G_u, \sigma'$); untill $g \neq null$ {
4)         $proj_1(DB, g) = \bigcup_{\langle id,d\rangle \in DB} proj_1(\langle id, d\rangle, g)$
5)         $F'$ = FTSMiner($proj_1(DB, g), \sigma'$)
6)         $F = F \cup \{\alpha \mid \alpha \in F' \wedge g_u(\alpha) = g\}$
7)     }
```

Fig. 4. Algorithm for mining rFTSs

Figure 4 shows an algorithm for enumerating all rFTSs F from DB. First, a set G_u of the union graphs of graph sequences DB is generated in Line 2. Assuming that the function call "FrequentSubgraphMiner" [8] repeatedly and exhaustively outputs connected frequent subgraphs g in G_u one at a time in Line 3, FTSMiner, which is the altered PrefixSpan [10,6], is called in Line 5 with the transformation sequences projected in Line 4 to mine rFTSs from $proj_1(DB, g)$. Finally, rFTSs mined from $proj_1(DB, g) = \bigcup_{\langle id,d\rangle \in DB} proj_1(\langle id, d\rangle, g)$, whose union graphs are isomorphic to g, are added to F in Line 6. These processes are continued until the connected frequent subgraph g is exhausted in FrequentSubgraphMiner. We have implemented FrequentSubgraphMiner using AcGM [8] which is one of the conventional Graph Mining methods.

4 Proposed Method: GTRACE2

Most of the computation time of GTRACE is used to run the altered PrefixSpan. The reason why the PrefixSpan used in GTRACE needs so much computation time is as follows. Let G_u and g be a set of union graphs of all $\langle id, d\rangle \in DB$ and a frequent connected subgraph mined by FrequentSubgraphMiner from G_u, respectively. The union graphs in G_u are often dense even if each interstate in graph sequences is sparse, since a union graph in G_u is generated by superimposing interstates in a graph sequence. In addition, since the union graph of a graph sequence is a graph with no labels, there exist many injective functions $V(g) \rightarrow V(g_u)$ between g and a dense union graph $g_u(d) \in G_u$ such that g is a subgraph of $g_u(d)$. Therefore, many projected transformation subsequences are produced from the graph g and one graph sequence d such that g is a subgraph of $g_u(d)$ according to the definition of projection.

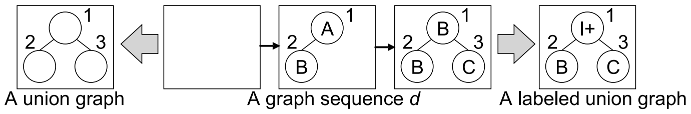


Fig. 5. Union Graph and Labeled Union Graph

To reduce the number of transformation subsequences produced by the projection, we redefine the union graph as follows:

Definition 7. *We redefine a union graph of d as $g_u(d) = (V_u, E_u, L \cup \{l_+\}, f_u)$ such that*

$$f_u(o)_{|o \in V_u} = \begin{cases} l & \textit{if always } f(v) = l \textit{ for } v \in V(g^{(j)}) \\ & \quad \textit{such that } g^{(j)} \in d \textit{ and } id(v) = o \\ l_+ & \textit{otherwise} \end{cases} \tag{3}$$

where V_u and E_u are given by Eqs. (1) and (2), respectively. L is a set of vertex labels in d, f is a function to assign vertex label $l \in L$ to each vertex in interstates in d, and $l_+ \notin L$. ■

The union graph defined here is a labeled graph, although the union graph defined in Section 3 is an unlabeled graph. So, we call the union graph we have defined here a labeled union graph. A label assigned to each vertex in the labeled union graph is determined by Eq. (3). If the vertices with unique ID o in interstates in d always have the identical label $l \in L$, a vertex o in the labeled union graph of d has the label l. Otherwise, the vertex o has a label l_+ such that $l_+ \notin L$.

Example 4. Figure 5 shows a union graph and a labeled union graph generated from the same graph sequence. Since two vertices with unique ID 1 in the graph sequence d have different labels, the corresponding vertex in the labeled union graph has a label l_+.

As mentioned in Section 3, GTRACE generates union graphs of all graph sequences in DB and mines all frequent connected subgraph patterns using AcGM. In this process, AcGM checks whether a pattern is included in each union graph. Since vertices with label l_+ in a labeled union graph should match any vertex in a pattern, the subgraph isomorphism test used in AcGM is altered as follows. Given two graphs $g(V, E, L, f)$ and $g'(V', E', L', f')$, g' is a subgraph of g, if there exists an injective function $\phi : V' \to V$ that satisfies the following conditions for $\forall v, v_1, v_2 \in V'$.

1. $(\phi(v_1), \phi(v_2)) \in E$, if $(v_1, v_2) \in E'$, and
2. $f(\phi(v)) = f'(v)$ or $f(\phi(v)) = l_+$.

By integrating the definition of the labeled union graph and the subgraph isomorphism test with GTRACE, we propose a new method called GTRACE2 to mine all rFTSs from graph sequences. According to the following lemma, we reduce the computation time to mine all rFTSs from graph sequences.

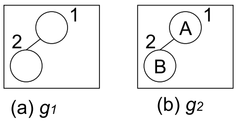


Fig. 6. Inputs of projection of GTRACE and GTRACE2

Lemma 1. *If g_1 is an unlabeled graph generated by removing all labels from a labeled graph g_2 to be used as input of projection in GTRACE2, then*

$$|\cup_{\langle id,d\rangle \in DB}\, proj_2(\langle id,d\rangle, g_2)| \leq |\cup_{\langle id,d\rangle \in DB}\, proj_1(\langle id,d\rangle, g_1)|,$$

where $proj_1$ and $proj_2$ are functions to project a graph sequence in GTRACE and GTRACE2, respectively. In addition, for $\langle id_2, seq(d_2)\rangle \in proj_2(\langle id_2, d\rangle, g_2)$, there must exists a transformation sequence $\langle id_1, seq(d_1)\rangle \in proj_1(\langle id_1, d\rangle, g_1)$ such that $id_1 = id_2$ and $seq(d_2) \sqsubseteq seq(d_1)$. Therefore, the average number of TRs in transformation sequences in $\cup_{\langle id,d\rangle \in DB} proj_2(\langle id,d\rangle, g_2)$ is less than or equal to the average number of TRs in transformation sequences in $\cup_{\langle id,d\rangle \in DB} proj_1(\langle id,d\rangle, g_1)$. ■

The proof of Lemma 1 is omitted due to the lack of space, but an example is given in Example 5. As shown in the experiments in [10], the computation time to run PrefixSpan is proportional to the number of sequences in its input, and it increases exponentially when the average number of items in the sequences increases. According to Lemma 1, since the number of transformation sequences generated by the projection in GTRACE2 usually decreases and the average number of TRs in the transformation sequences usually becomes less than in the original GTRACE, the computation time for running the altered PrefixSpan in GTRACE2 is reduced.

Example 5. The graph sequence $\langle 1, d\rangle$ at the center of Fig. 5 is represented as $\langle 1, \langle vi^{(1,1)}_{[1,A]} vi^{(1,2)}_{[2,B]} ei^{(1,3)}_{[(1,2),\bullet]} vi^{(2,1)}_{[3,C]} ei^{(2,2)}_{[(1,3),\bullet]} vr^{(2,3)}_{[1,B]}\rangle\rangle$, and its union graph and labeled union graph are shown in Fig. 5. Given the graph g_1 shown in Fig. 6 (a) as input of the projection in GTRACE, two vertices in g_1 correspond to vertices with unique IDs 1 and 2 or vertices with unique IDs 1 and 3 in the union graph $g_u(d)$. Therefore, $proj_1(\langle 1, d\rangle, g_1)$ is

$$\{\langle 1, \langle vi^{(1,1)}_{[1,A]} vi^{(1,2)}_{[2,B]} ei^{(1,3)}_{[(1,2),\bullet]} vr^{(2,3)}_{[1,B]}\rangle\rangle, \langle 1, \langle vi^{(1,1)}_{[1,A]} vi^{(2,1)}_{[3,C]} ei^{(2,2)}_{[(1,3),\bullet]} vr^{(2,3)}_{[1,B]}\rangle\rangle\}.$$

On the other hand, given the graph g_2 shown in Fig. 6 (b) as input of the projection in GTRACE2, $proj_2(\langle 1, d\rangle, g_2)$ is

$$\{\langle 1, \langle vi^{(1,1)}_{[1,A]} vi^{(1,2)}_{[2,B]} ei^{(1,3)}_{[(1,2),\bullet]}\rangle\rangle\},$$

since two vertices with unique ID 1 and 2 in the input graph g_2 correspond to vertices with unique IDs 1 and 2 in the labeled union graph $g_u(d)$, respectively. This projected transformation sequence does not include $vr^{(2,3)}_{[1,B]}$ to satisfy $g_u(d) = g$ in Definition 8.

Table 2. Results for the Enron dataset

$\lvert ID_V(d) \rvert$	80	90	100	110	120	140	150	170	182
GTRACE comp. time	0.18	42.0	1509.5	–	–	–	–	–	–
comp. time for PrefixSpan	0.048	37.7	1407.7	–	–	–	–	–	–
# of subgraph patterns	4	7	10	–	–	–	–	–	–
avg. # of trans. seq.	809	19249	269726	–	–	–	–	–	–
avg. # of TRs	10.3	23.4	30.1	–	–	–	–	–	–
GTRACE2 comp. time	0.14	0.25	0.34	0.39	0.63	0.67	1.2	3.8	4.1
comp. time for PrefixSpan	0.015	0.062	0.078	0.11	0.22	0.20	0.45	2.5	2.7
# of subgraph patterns	22	26	28	29	36	39	45	50	52
avg. # of trans. seq.	197	415	585	656	813	892	1333	2078	2201
avg. # of TRs	4.7	6.2	6.5	6.4	6.7	6.7	7.0	7.9	7.9
σ', n	60%	50%	40%	20%	5%	2%	5	6	7
GTRACE comp. time	7.5	132.4	–	–	–	–	–	–	–
comp. time for PrefixSpan	1.7	60.9	–	–	–	–	–	–	–
# of subgraph patterns	4	7	–	–	–	–	–	–	–
avg. # of trans. seq.	41334	202432	–	–	–	–	–	–	–
avg. # of TRs	16.8	22.3	–	–	–	–	–	–	–
GTRACE2 comp. time	0.52	0.92	1.2	2.7	25.6	327.3	1.3	2.2	4.1
comp. time for PrefixSpan	0.03	0.11	0.17	1.4	24.0	325.5	0.50	0.97	2.7
# of subgraph patterns	13	24	29	46	81	124	44	47	52
avg. # of trans. seq.	3288	2906	3028	2424	1468	991	1940	2293	2201
avg. # of TRs	4.7	6.0	6.6	7.7	8.2	8.7	6.0	6.7	7.9

comp. time: computation time [sec],
comp. time for PrefixSpan: total computation time to run the altered PrefixSpan,
# of subgraph patterns: the number of frequent connected subgraphs mined by AcGM,
avg. # of trans. seq.: the average number of transformation sequences in $proj_i(DB, g)$,
avg. # of TRs: the average number of TRs in transformation sequences in $proj_i(DB, g)$
Default: minimum support σ' =15%, # of vertex labels $|L_v| = 8$, # of edge labels $|L_e| = 1$,
# of persons $|ID_V(d)| = 182$, # of interstates in a graph sequence n=7.

5 Experiment and Discussion

The proposed method was implemented in C++. The experiments were executed on an HP xw4600 with an Intel Core 2 8600 3.33 GHz processor and 2 GB of main memory and running Windows XP. The performance of the proposed method was evaluated using both artificial and real world graph sequence data. Due to the lack of space, we report the experiments using the real world data.

To assess the practicality of the proposed method, it was applied to the Enron Email Dataset [3]. We assigned a unique ID to each person participating in email communication, and assigned an edge to a pair communicating via email on a particular day, thereby obtaining a daily graph $g^{(j)}$. In addition, one of the vertex labels {CEO, Employee, Director, Manager, Lawyer, President, Trader, Vice President} was assigned to each vertex. We then obtained a set of weekly graph sequence data DB. The total number of weeks, *i.e.*, number of sequences, was 123. We randomly sampled $|ID_V(d)|(= 1 \sim 182)$ persons to form DB.

Table 2 shows the computation times [sec] to run GTRACE and GTRACE2, total computation times [sec] to run the altered PrefixSpan, the numbers of frequent connected subgraphs mined by AcGM, the average numbers of transformation sequences in $proj_i(DB, g)$, and the average numbers of transformation sequences in $proj_i(DB, g)$ obtained for various numbers of unique IDs (persons) $|ID_V(d)|$, minimum support σ', and numbers of interstates n in each graph sequence of the dataset. All the other parameters were set to the default values indicated at the bottom of the table. Thus, the dataset with the default values as contained 123 graph sequences each consisting of 182 persons (unique IDs) and 7 interstates. The parameter n =5, 6, or 7 indicates that each sequence d in DB consists of 5, 6, or 7 steps (interstates) from Monday to Friday, Saturday, or Sunday, respectively. When the required computation time exceeds two hours or a memory overflow occurs, the results are indicated by "-".

The upper, lower left, and lower right parts of the table show experimental results with regard to the number of persons (unique IDs), the minimum support threshold, and the number of interstates in graph sequences in the graph sequence database, respectively. The table indicates that GTRACE proved intractable for the graph sequence dataset generated from the default values, althought the proposed method GTRACE2 is tractable with respect to the database. In addition, the computation times for both GTRACE and GTRACE2 are exponential with respect to the increases of the number of $|ID_V(d)|$ and the number of interstates in the graph sequence database and with respect to the decrease of the minimum support threshold. The main reason that the computation time increases is the increase in the number of frequent patterns.

The computation times for GTRACE2 are much smaller than those for GTRACE, although the number of times to call the altered PrefixSpan increases. Most of computation time of GTRACE is used running the altered PrefixSpan. As shown in [10], the computation time of PrefixSpan is proportional to the number of sequences in its input and increase exponentially with respect to the average number of items in sequences in its input. The scalability of GTRACE2 comes from reducing the number of transformation sequences and the number of TRs in transformation sequences in the projected database by using the labeled union graph proposed in Section 4. GTRACE2 is practical, because it can be applied to graph sequences that are both longer and larger than those to which GTRACE can be applied.

6 Conclusion

In this paper, we proposed a method to improve GTRACE which mines a set of relevant frequent transformation subsequences (rFTSs) from given graph sequences by defining the labeled union graph. We developed a graph sequence mining program GTRACE2, and confirmed its efficiency and practical performance through computational experiments using artificial and real world datasets. Our performance study showed that the proposed method is some orders of magnitude faster than the conventional method, and is efficient and scalable for mining both long and large graph sequence patterns. Recently, we have proposed a method for mining another class of frequent patterns, called FRISSs (Frequent, Relevant, and Induced Subgraph Subsequences), from graph sequence [7]. The principle proposed in this paper using labeled graphs can be applied to the method. In real applications, it is hard to enumerate usefule and interesting FTSs which are exactly included in graph sequences. In future work, we plan to extend GTRACE2 to mine FTSs which are approximately included in graph sequences by using sliding windows and time constraints proposed in [11].

References

1. Berlingerio, M., et al.: Mining Graph Evolution Rules. In: Proc. of Euro. Conf. on Principles and Practice of Knowledge Discovery in Databases, pp. 115–130 (2009)
2. Borgwardt, K.M., et al.: Pattern Mining in Frequent Dynamic Subgraphs. In: Proc. of Int'l Conf. on Data Mining, pp. 818–822 (2006)
3. Enron Email Dataset, `http://www.cs.cmu.edu/~enron/`
4. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (1979)
5. Inokuchi, A., et al.: An Apriori-based Algorithm for Mining Frequent Substructures from Graph Data. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 13–23. Springer, Heidelberg (2000)
6. Inokuchi, A., Washio, T.: A Fast Method to Mine Frequent Subsequences from Graph Sequence Data. In: Proc. of Int'l Conf. on Data Mining, pp. 303–312 (2008)
7. Inokuchi, A., Washio, T.: Mining Frequent Graph Sequence Patterns Induced by Vertices. In: Proc. of SIAM Int'l Conf. on Data Mining (2010)
8. Inokuchi, A., et al.: A Fast Algorithm for Mining Frequent Connected Subgraphs. IBM Research Report, RT0448 (2002)
9. Nijssen, S., Kok, J.N.: A Quickstart in Frequent Structure Mining can Make a Difference. In: Proc. of Int'l Conf. on Knowledge Discovery and Data Mining, pp. 647–652 (2004)
10. Pei, J., et al.: PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth. In: Proc. of Int'l Conf. on Data Eng., pp. 2–6 (2001)
11. Srikant, R., Agrawal, R.: Mining Sequential Patterns: Generalizations and Performance Improvements. In: Proc. of Int'l Conf. on Extending Database Technology, pp. 3–17 (1996)
12. Yan, X., Han, J.: gSpan: Graph-Based Substructure Pattern Mining. In: Proc. of Int'l Conf. on Data Mining, pp. 721–724 (2002)

Orthogonal Nonnegative Matrix Tri-factorization for Semi-supervised Document Co-clustering

Huifang Ma, Weizhong Zhao, Qing Tan, and Zhongzhi Shi

Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, 100190 Beijing, China
Graduate University of the Chinese Academy of Sciences, 100049 Beijing, China
mahf@ics.ict.ac.cn
http://www.intsci.ac.cn/users/mahuifang/index.html

Abstract. Semi-supervised clustering is often viewed as using labeled data to aid the clustering process. However, existing algorithms fail to consider dual constraints between data points (e.g. documents) and features (e.g. words). To address this problem, in this paper, we propose a novel semi-supervised document co-clustering model OSS-NMF via orthogonal nonnegative matrix tri-factorization. Our model incorporates prior knowledge both on document and word side to aid the new word-category and document-cluster matrices construction. Besides, we prove the correctness and convergence of our model to demonstrate its mathematical rigorous. Our experimental evaluations show that the proposed document clustering model presents remarkable performance improvements with certain constraints.

Keywords: Semi-supervised Clustering, Pairwise Constraints, Word-Level Constraints, Nonnegative Matrix tri-Factorization.

1 Introduction

Providing a meaningful cluster hierarchy to a document corpus has always been a major goal for the data mining community. Approaches to solve this problem have focused on document clustering algorithms, which are widely used in a number of different areas of text mining and information retrieval. One of a latest presented approach for obtaining document cluster is Non-negative Matrix Factorization (NMF) [1], which aimed to provide a minimum error non-negative representation of the term-document matrix. This technique can be considered as co-clustering [2], which aimed to cluster both the rows and columns of the original data simultaneously by making efficient use of the duality between data points (e.g. documents) and features (e.g. words). Put it another way, document clustering and word clustering are performed in a reinforcing manner.

However, traditional clustering algorithms fail to take advantage of knowledge from domain experts. Incorporating the additional information can greatly enhance the performance of clustering algorithms. In recent years, a great amount of effort has been made for clustering document corpus in a semi-supervised way, aiming to cluster the document set under the guidance of some supervisory information.

Unfortunately, traditional approaches to semi-supervised document clustering inherently strongly depend on constraints within document themselves while ignore the useful semantic correlation information hidden within the words of the document corpus. We believe that adding word semantic information (such as word clusters indicating word semantics) as additional constraints can definitely improve document clustering performance. Thereafter how to effectively combine both document-level and word-level constraints to guide the process of document clustering is a problem that is definitely worthy of researching.

Based on the above considerations, in this paper, we propose a novel semi-supervised document co-clustering method via non-negative factorization of the term-document matrix for the given document corpus. We have extended the classical NMF approach by introducing both document-level and word-level constraints based on some prior knowledge. Our clustering model encodes the user's prior knowledge with a set of constraints to the objective function, and the document clustering task is carried out by solving a constrained optimization problem. Specifically, we propose a semi-supervised co-clustering framework to cluster the words and documents simultaneously. Meanwhile, we derive iterative algorithms to perform orthogonal non-negative tri-factorization. The correctness and convergence of these algorithms are proved by showing that the solution satisfied the KKT optimality and these algorithms are guaranteed to converge. Experiments performed on various publicly available document datasets demonstrate the superior performance of the proposed work.

The basic outline of this paper is as follows: Section 2 introduces related works. Section 3 presents the semi-supervised orthogonal nonnegative matrix tri-factorization. The experiments and results are given in Section 4. Lastly, we conclude our paper in Section 5.

2 Related Work

This section briefly reviews related work about NMF and semi-supervised document clustering.

The classical NMF algorithms [3] aim to find two matrix factors for a matrix X such that $X \approx WH^T$, where $W^{m \times k}$ and $H^{n \times k}$ are both nonnegative matrices. Ding et al.[4] made systematical analysis of NMF and introduced 3-factor NMF. They demonstrated that the orthogonality constraint leads to rigorous clustering interpretation. When 3-factor NMF is applied to the term-document matrix X, each column X_j of X is an encoding of an original document and each entry x_{ij} of vector X_j is the significance of term i with respect to the semantics of X_j, where i ranges across the terms in the dictionary. Thereafter, Orthogonal NMF factorizes X into three non-negative matrices

$$X = FSG^T, \tag{1}$$

where G is the cluster indicator matrix for clustering of documents of X and F is the word cluster indicator matrix for clustering of rows of X. The simultaneous row/column clustering can be solved by optimizing

$$J = \min_{F\geq 0, S\geq 0, G\geq 0} \left\| X - FSG^T \right\|_F^2 \quad s.t \quad F^TF = I, G^TG = I. \tag{2}$$

The Frobenius norm is often used to measure the error between the original matrix X and its low rank approximation FSG^T. The rank of the approximation, k, is a parameter that must be set by users.

Several formulations of co-clustering problem are proposed in the past decade and they are superior to traditional one-side clustering. Dhillon [2] proposed a bipartite spectral graph partitioning approach to co-cluster words and documents. Long et al.[5] presented a general principled model, called relation summary network to co-cluster the heterogeneous data on a k-partite graph. As for semi-supervised co-clustering algorithms, Chen et al.[6] presented a semi-supervised document clustering model with simultaneous text representation and categorization. Fei et al.[7] proposed a semi-supervised clustering algorithm via matrix factorization. Li et al.[8] presented an interesting word-constrained clustering algorithm. The way of incorporating word constraints is very appealing and sets a good foundation for our model formulation. Even though these semi-supervised algorithms have shown to be superior to traditional clustering method, very little is known about the combination of constraints on both documents and words. One recent work came from Li et al.[9]. They have demonstrated a non-negative matrix tri-factorization approach to sentiment classification with prior knowledge about sentiment words in the lexicon and partial labels on documents.

3 Semi-supervised Orthogonal Nonnegative Matrix Tri-factorization for Co-clustering

In this section, we first describe how we integrate two different constraints in our model in Sect. 3.1. We then derive the OSS-NMF model, prove the correctness and convergence of the algorithm in Sect. 3.2 and Sect. 3.3 respectively.

3.1 Incorporating Document-Level Constraints

Our model treats the prior knowledge on the word side as categorization of words, represented by a complete specification F_0 for F. The prior knowledge on document-level is provided in the form of two sets of pairwise constraints on documents: two documents are known to be highly related and must be grouped into the same document cluster; or two documents that are irrelevant and can not be grouped into the same cluster.

We make use of set A_{ml} to denote that must-link document pairs (d_{i_1}, d_{j_1}) are similar and must be clustered into the same document cluster:

$$A_{ml} = \{(i_1; j_1); \ldots; (i_a; j_a)\}; a = |A_{ml}|. \tag{3}$$

It is easy to demonstrate that the must-link constraints represent equivalence relation. Therefore, we can compute a collection of transitive closures from A_{ml}. Each pair of documents in the same transitive closure must be in the same cluster in the clustering result.

Meanwhile, cannot-link document pairs are collected into another set:

$$B_{cl} = \{(i_1; j_1); \ldots ; (i_b; j_b)\}; b = |B_{cl}|, \tag{4}$$

where each pair of documents are considered dissimilar and ought not to be clustered into the same document cluster.

We then encode the must-link document pairs as a symmetric matrix A whose diagonal entries are all equal to one and the cannot-link document pairs as another matrix B.

Suppose each document in the corpus either completely belongs to a particular topic, or is more or less related to several topics. We can then regard these constraints as the document class posterior probability on G. A must-link pair $(i_1; j_1)$ implies that the overlap $g_{i_1k}g_{j_1k} > 0$ for some class k, and therefore $\sum_k g_{i_1k}g_{j_1k} = (GG^T)_{i_1j_1}$ should be maximized. The must-link condition can be presented as

$$\max_G \sum_{i,j\in A} (GG^T)_{ij} = \sum_{ij} A_{ij}(GG^T)_{ij} = TrG^T AG. \tag{5}$$

In terms of cannot-link pairs $(i_2; j_2)$, $g_{i_2k}g_{j_2k} = 0$ for all k. Likewise, we take the cannot-link constraints and minimize $\sum_k g_{i_2k}g_{j_2k} = (G^T G)_{i_2j_2}$. Since g_{ik} are nonnegative, we write this condition as:

$$\sum_{i,j\in B} (GG^T)_{ij} = TrBGG^T = 0, \quad or \quad \min_G TrG^T BG. \tag{6}$$

3.2 Algorithm Derivation

Combining the above constraints together, we define the objective function of OSS-NMF as:

$$J = \min_{F\geq 0, S\geq 0, G\geq 0} ||X - FSG^T|| \; + \; \alpha \, \|F - F_0\|_F^2 + Tr(-\beta GAG^T + \gamma GBG^T),$$
$$s.t. \; FF^T = I, GG^T = I, \tag{7}$$

where α, β and γ are positive trade-off parameters that control the degree of enforcement of the user's prior knowledge. The larger value the parameters take, the stronger enforcement of the users prior knowledge we will have; vise versa.

An iterative procedure to solve the optimization problem in Eq.(7) can be summarized as follows.

Computation of *S*. Optimizing Eq.(7) with respect to S is equivalent to optimizing

$$J_1 = \min_{F\geq 0, S\geq 0, G\geq 0} \left\| X - FSG^T \right\|_F^2. \tag{8}$$

Setting $\frac{\partial J_1}{\partial S} = 0$ leads to the following updating formula:

$$S_{ik} = S_{ik} \sqrt{\frac{(F^T XG)_{ik}}{(F^T FSG^T G)_{ik}}}. \tag{9}$$

Computation of *F*. Optimizing Eq.(7) with respect to F is equivalent to optimizing

$$J_2 = \min_{F\geq 0, S\geq 0, G\geq 0} \left\|X - FSG^T\right\|_F^2 + \alpha \left\|F - F_0\right\|_F^2, \quad s.t. \quad FF^T = I. \tag{10}$$

We present an iterative multiplicative updating solution. After introducing the Lagrangian multiplier, the Lagrangian function is stated as

$$L(F) = \left\|X - FSG^T\right\|_F^2 + \alpha \left\|F - F_0\right\|_F^2 + Tr[\lambda_1(F^T F - I)]. \tag{11}$$

This takes the exact form as Li demonstrated in [8], thereby we can update F as follows:

$$F_{ik} = F_{ik}\sqrt{\frac{(XGS^T + \alpha F_0)_{ik}}{(FF^T XGS^T + \alpha FF^T F_0)_{ik}}}. \tag{12}$$

Computation of *G*. Optimizing Eq.(7) with respect to G is equivalent to optimizing

$$J_3 = \min_{F\geq 0, S\geq 0, G\geq 0} \left\|X - FSG^T\right\|_F^2 + Tr(-\beta G^T AG + \gamma G^T BG), \quad s.t. \quad GG^T = I. \tag{13}$$

Similar with the computation of F, we introduce the Lagrangian multiplier, thus the Lagrangian function is

$$L(G) = \left\|X - FSG^T\right\|^2 + Tr(-\beta G^T AG + \gamma G^T BG) + Tr[\lambda_2(G^T G - I)]. \tag{14}$$

We show that G can be iterated as:

$$G_{ik} = G_{ik}\sqrt{\frac{(X^T FS + \beta AG)_{ik}}{(G(SF^T FS^T + \lambda_2) + \gamma BG)_{ik}}}. \tag{15}$$

The detailed analysis of computation of G is shown in the optimization section. When the iteration starts, we update one factor with others fixed.

3.3 Algorithm Correctness and Convergence

To prove the correctness and convergence of our algorithm, we will make use of optimization theory, matrix inequalities and auxiliary functions that used in [3].

Correctness

Theorem 1. *If the update rule of S, F and G in Eq.(9), Eq.(12) and Eq.(15) converge, then the final solution satisfies the KKT optimality condition, i.e., the algorithm converges correctly to a local optima.*

Proof: Following the standard theory of constrained optimization, we introduce the Lagrangian multipliers λ_1, λ_2 and construct the following Lagrangian function:

$$\begin{aligned} L &= \|X - FSG^T\| + \alpha\|F - F_0\| + Tr[\lambda_1(F^TF - I)] \\ &\quad + Tr[-\beta GAG^T + \gamma GBG^T + \lambda_2(G^TG - I)] \\ &= Tr[X^TX - 2G^TX^TFS + G^TGSF^TFS + \alpha(FF^T - 2FF_0^T + F_0F_0^T)] \\ &\quad - \beta G^TAG + \gamma G^TBG + \lambda_1(F^TF - I) + \lambda_2(G^TG - I)]. \end{aligned} \tag{16}$$

The correctness of updating rules for S in Eq.(9) and F in Eq.(12) have been proved in [8]. Therefore, we only need to proof the correctness of updating rules for G. Fixing F, S, we can get that the KKT complementary condition for the non-negativity of G

$$[-2X^TFS + 2G(SF^TFS^T + \lambda_2) - 2\beta AG + 2\gamma BG]_{ik}G_{ik} = 0. \tag{17}$$

We then obtain the Lagrangian multiplier, it is obvious that at convergence the solution satisfy

$$[-2X^TFS + 2G(SF^TFS^T + \lambda_2) - 2\beta AG + 2\gamma BG]_{ik}G_{ik}^2 = 0. \tag{18}$$

We can see that this is identical to the KKT condition. The above equation denotes that either the first factor equals to zero, or G_{ik} is zero. If the first factor is zero, the two equations are identical. If G_{ik} is zero, then G_{ik}^2 is zero as well, vice versa. Thus, we have proved that if the iteration converges, the converged solution satisfies the KKT condition, i.e., it converges correctly to a local minima.

Proof is completed.

Convergence. We demonstrate that the above objective function decreased monotonically under these three updating rules. Before we proof the convergence of the algorithm, we need to construct the auxiliary function similar to that used in Lee and Seung [3]. We first introduce the definition of auxiliary function.

Definition 1. *A function $Z(H, H')$ is called an auxiliary function of $L(H)$ if it satisfies*

$$Z(H, H') \geq L(H), \quad Z(H, H) = L(H). \tag{19}$$

Lemma 1. *If $Z(H, H')$ is an auxiliary function, then L is non-increasing under the update*

$$H^{(t+1)} = \arg\min_H Z(H, H^t). \tag{20}$$

By construction $L(H^{(t)}) = Z(H^{(t)}, H^{(t)}) \geq Z(H^{(t+1)}, H^{(t)}) \geq L(H^{(t+1)})$, $L(H^{(t+1)})$ is monotonic decreasing (non-increasing).

Lemma 2. *For any nonnegative matrices $A \in R^{n\times n}$,$B \in R^{k\times k}$,$S \in R^{n\times k}$,$S' \in R^{n\times k}$, A, B are symmetric, the following inequality holds[10]:*

$$\sum_{i=1}^{n}\sum_{p=1}^{k} \frac{(AS'B)_{ip}S_{ip}^2}{S'_{ip}} \geq tr(S^TASB). \tag{21}$$

Theorem 2. *The above iterative algorithms converge.*

Proof: To proof the algorithm converges, the key step is to find an appropriate auxiliary function $Z(G, G')$ of L(G) in Eq.(14). We show that the following function

$$Z(G,G') = \sum_{ik} [-2G'_{ik}(1+\log\frac{G_{ik}}{G'_{ik}})(X^TFS)_{ik} + \frac{[G'(SF^TFS+\lambda_2)]_{ik}G^2_{ik}}{G'_{ik}} \\ - \beta G'_{ik}(AG')_{ik}(1+\log\frac{G^2_{ik}}{G'_{ik}}) + \gamma\frac{(BG')_{ik}G^2_{ik}}{G'_{ik}}]. \quad (22)$$

is its corresponding auxiliary function.

First, it is obvious that when $G = G'$, the equality holds. Second, the inequality holds $Z(G, G') \geq L'(G)$. This is based on the following: a) The first term and third term in $Z(G, G')$ are always smaller than the corresponding terms in $L'(G)$ because of the inequality $z \geq 1 + \log(z) \quad \forall z > 0$; b) The second and last term in Eq.(24) are always bigger than the corresponding terms in $L'(G)$, due to Lemma 2. Putting these together, we can guarantee that $Z(G, G') \geq L'(G)$.

To find the minimum of $Z(G, G')$, we take

$$\frac{\partial Z(G,G')}{\partial G_{ik}} = \sum_{ik} [-2\frac{G'_{ik}}{G_{ik}}(X^TFS)_{ik} + 2\frac{[G'(SF^TFS+\lambda_2)]_{ik}G_{ik}}{G'_{ik}} \\ - 2\beta\frac{G'_{ik}(AG')_{ik}}{G_{ik}} + 2\gamma\frac{(BG')_{ik}G_{ik}}{G'_{ik}}] \quad (23)$$

and the Hessian matrix of $Z(G, G')$

$$\frac{\partial^2 Z(G,G')}{\partial G_{ik}\partial G_{jl}} = \sum_{ik} [2\frac{G'_{ik}}{G^2_{ik}}(X^TFS)_{ik} + 2\frac{[G'(SF^TFS+\lambda_2)]_{ik}}{G'_{ik}} \\ + 2\beta\frac{G'_{ik}(AG')_{ik}}{G^2_{ik}} + 2\gamma\frac{(BG')_{ik}}{G'_{ik}}]\delta_{ij}\delta_{kl} \quad (24)$$

is a diagonal matrix with positive diagonal elements.

Thus $Z(G, G')$ is a convex function of G. Therefore, we can obtain the global minimum of Z. The minimum value is obtained by setting $\frac{\partial Z(G,G')}{\partial G_{ik}} = 0$, we get

$$\frac{G'_{ik}}{G_{ik}}(X^TFS+\beta AG)_{ik} = \frac{G_{ik}}{G'_{ik}}(G'(SF^TFS^T)+\lambda_2+\gamma BG')_{ik}. \quad (25)$$

We can thereafter derive the updating rule of Eq.(16)

$$G_{ik} = G_{ik}\sqrt{\frac{(X^TFS+\beta AG)_{ik}}{(G(SF^TFS^T+\lambda_2)+\gamma BG)_{ik}}}. \quad (26)$$

Under this updating rule, $L'(G)$ decreases monotonically, where the Lagrangian multiplier k-by-k matrix λ_2 for enforcing the orthogonality and $G^TG = I$ is given by

$$\lambda_2 = G^TX^TFS + \beta G^TAG - \gamma G^TBG - SF^TFS^T. \quad (27)$$

Proof is completed.

4 Experiments

This section provides empirical evidence to show the benefits of our model OSS-NMF. We compared our method with Constrained-Kmeans[11], Information-Theoretic Co-clustering, which is referred to as IT-Co-clustering[12], ONMF-W denoting Orthogonal NMF with word-level constraints[8], ONMF-D representing Orthogonal NMF with document-level constraints. Constrained K-means is the representative semi-supervised data clustering method; Information-Theoretic Co-clustering is one of the most popular co-clustering method; ONMF-W and ONMF-D are two derived algorithms from our approach.

The requirement of word constraints is the specification of word categorization. Similar with Li [8], we took advantage of the ACM term taxonomy, which come naturally and strictly decide the taxonomy of computer society. The document-level constraints were generated by randomly selecting pairs of documents. If the labels of this document pair are the same, then we generated a must link. In contrast, if the labels are different, a cannot link is generated. The amounts of constraints were determined by the size of input data. Incorporating dual constraints on our model, we believe that our approach should perform better given reasonable amount of labeled data.

4.1 Datasets

Three different datasets widely used as benchmark data sets in clustering literature were used.

Citeseer dataset: Citeseer collection was made publicly available by Lise Getoor's research group at University of Maryland. We end up with a sampling of Citeseer data containing 3312 documents. These data are classified into one of the following six classes: Agents, Artificial Intelligence, Data Base, Information Retrieval, Machine Learning, Human Computer Interaction.

DBLP Dataset: This dataset is downloaded from DBLP Computer Science Bibliography spanning from 1999 to 2004. We extract the paper titles to form our dataset from 5 categories, which contains 2170 documents.

URCS Technical Reports: This dataset is composed of abstracts of technical reports published in the Department of Computer Science at Rochester University. There are altogether 512 reports abstracts grouped according to 4 categories.

We pre-processed each document by tokenizing the text into bag-of-words. Then we applied stopwords removing and stemmed words. In particular, words that occur in less than three documents are removed. We used the weighted term-frequency vector to represent each document.

4.2 Evaluation Metrics

We adopt the clustering accuracy and normalized mutual information as our performance measures. These performance measures are standard measures widely used for clustering. Clustering accuracy measures the cluster performance from the one-to-one relationship between clusters and classes point of view, which is defined as:

$$Acc = \max \frac{\sum_{i=1}^{N} \delta(map(r_i), d_i)}{N}, \tag{28}$$

where r_i denotes the cluster label of a document and d_i denotes the true class label, N is the total number of documents, $\delta(x, y)$ is a function which equals one if $x = y$ and equals zero otherwise, $map(r_i)$ is the permutation function which maps each cluster label to the corresponding label of the data set.

NMI measures how closely the clustering algorithm could reconstruct the underlying label distribution in the data. It is defined as:

$$NMI = \frac{I(Z'; Z)}{(H(Z') + H(Z))/2}, \tag{29}$$

where $I(Z'; Z) = H(Z) - H(Z|Z')$ is the mutual information between the random variables Z' and Z, $H(Z)$ is the Shannon entropy of Z, and $H(Z|Z')$ is the conditional entropy of Z given Z'. In general, the larger the NMI value is, the better the clustering quality is.

4.3 Clustering Results

Considering the document constraints are generated randomly, we run each algorithm 20 times for each dataset and took the average as statistical results. To give these algorithms some advantage, we set the number of clusters equal to the real number of all the document clusters and word clusters.

Overall Evaluation. Table 1 shows the cluster accuracy and normalized mutual information of all the algorithms on all the data sets. From the experimental comparisons, we observe that our proposed method OO-SNMF effectively combined prior knowledge from the word side with constraints on the document side for improving clustering results. Moreover, our model outperforms most of the clustering methods on all the data sets. In summary, the experimental results match favorably with our hypotheses and encouraged us to further explore the reasons.

The superiority of our model arises in the following three aspects: (1) the mechanism of tri-factorization for term-document matrix allows setting different classes of terms and documents, which is in line with the real applications; (2) co-clustering the terms and documents with both constraints leads to improvement in the clustering of documents; (3) last but not least, the constraints on word-level are quite different from that of document-level, which means our model can incorporate distinguished semantic information on both sides for clustering.

Effect of the Size of Words. In this section, we describe the effect of the size of words on clustering. These words can be used to represent the underlying

Table 1. Comparison of four algorithms on different datasets

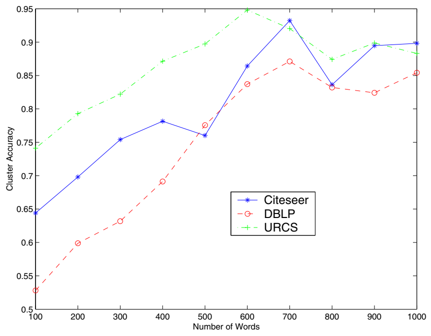
(a) Clustering Accuracy

Data Sets	Citeseer	DBLP	URCS
Constrained-Kmeans	0.5124	0.4215	0.5923
IT-Co-clustering	0.5765	0.4873	0.6214
ONMF-W	0.5514	0.4812	0.6052
ONMF-D	0.6142	0.5321	0.6812
OSS-NMF	**0.7235**	**0.6823**	**0.8368**

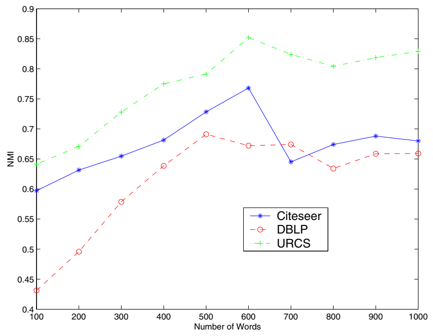
(b) Normalized Mutual Information

Data Sets	Citeseer	DBLP	URCS
Constrained-Kmeans	0.5813	0.5312	0.6358
IT-Co-clustering	0.6521	0.5821	0.7389
ONMF-W	0.6722	0.6312	0.7548
ONMF-D	0.7214	0.6523	0.7964
OSS-NMF	**0.8345**	**0.7643**	**0.9124**

'concept' of the corresponding category cluster. We follow the term frequency criteria to select word. The performance results with different numbers of words on all of the datasets are demonstrated.



(a) Cluster Accuracy with different numbers of words on 3 dataset.



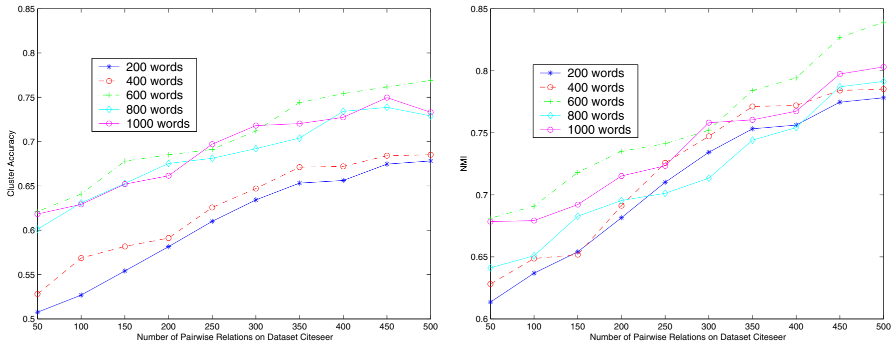
(b) NMI with different numbers of words on 3 dataset.

Fig. 1. Accuracy and NMI results with different numbers of words on 3 dataset

Both Accuracy and NMI show clear benefits of having more words: the performance increases as the amount of words grows, as shown in Fig.1. This indicates the addition of word semantic information can greatly help the clustering performance. It also shows a great variation with the increase of words. When the size of words increases beyond a certain value, the quality of clustering fluctuates and suddenly drops and then becomes stable.

Experiments on Pairwise Constraint of Documents. We conducted experiments for our framework by varying the number of pairwise constraints and size of words. Results from all these document collections indicate that generally as more and more constraints are added to the dataset being clustered, the performance of the clustering method becomes better, confirming previous discussion on the effect of increase of more labeled data. Due to the limitation of this paper, we only present NMI and Cluster Accuracy on Citeseer in Fig.2.

Our finding can be summarized as follows: (1) As long as the constraints are provided, our model always outperforms the traditional constrained methods. (2) The model performs much better with the increase of constraints.



(a) Cluster Accuracy on Citeseer (b) NMI results on Citeseer

Fig. 2. Accuracy and NMI results with different numbers of words and pairwise documents on Citeseer

5 Conclusions and Future Work

In this paper, we consider the problem of semi-supervised document co-clustering. We have presented a novel orthogonal semi-supervised nonnegative matrix tri-factorization model. We also have provided theoretical analysis of the correctness and convergence of the algorithm. The ability of our proposed algorithm to integrate double constraints makes it efficient for document co-clustering.

Our work leads to several questions. We incorporated the word prior knowledge as a specification of the initial word cluster. It would also be interesting to make use of pairwise constraints on the word side. In particular, a further interesting direction is to actively select informative document pairs for obtaining user judgments so that the clustering performance can be improved with as few supervised data as possible.

Acknowledgments. This work is supported by the National Science Foundation of China (No. 60933004, 60903141, 60775035), the National Basic Research Priorities Programme (No. 2007CB311004), 863 National High-Tech Program (No.2007AA-01Z132), and National Science and Technology Support Plan (No.2006BAC08B06).

References

1. Xu, W., Liu, X., Gong, Y.: Document Clustering Based on Non-negative Matrix Factorization. In: Proceedings of the 26th ACM SIGIR Conference on Research and Development in Information Retrieval, Toronto, Canada, pp. 267–273 (2003)
2. Dhillon, I.S.: Co-clustering Documents and Words Using Bipartite Spectral Graph Partitioning. In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, California, USA, pp. 269–274 (2001)

3. Lee, D., Seung, H.S.: Algorithms for Non-negative Matrix Factorization. In: Proceedings of 15th Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, vol. 13, pp. 556–562 (2001)
4. Ding, C., Li, T., Peng, W., Park, H.: Orthogonal Nonnegative Matrix Tri-factorizations for Clustering. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, pp. 126–135 (2006)
5. Long, B., Zhang, Z., Wu, X., Yu, P.S.: Spectral Clustering for Multi-type Relational Data. In: Proceedings of 23rd International Conference on Machine Learning, Pittsburgh, Pennsylvania, USA, pp. 585–592 (2006)
6. Chen, Y.H., Wang, L.J., Dong, M.: Semi-supervised Document Clustering with Simultaneous Text Representation and Categorization. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009. LNCS (LNAI), vol. 5781, pp. 211–226. Springer, Heidelberg (2009)
7. Wang, F., Li, T., Zhang, C.S.: Semi-Supervised Clustering via Matrix Factorization. In: Proceedings of The 8th SIAM Conference on Data Mining, Atlanta, Geogia, pp. 1–12 (2008)
8. Li, T., Ding, C., Zhang, Y., Shao, B.: Knowledge Transformation from Word Space to Document Space. In: Proceedings of the 31st Annual International ACM SIGIR conference on research and development in information retrieval, Singapore, pp. 187–194 (2008)
9. Li, T., Zhang, Y., Sindhwani, W.: A Non-negative Matrix Tri-factorization Approach to Sentiment Classification with Lexical Prior Knowledge. In: Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, Suntec, Singapore, pp. 244–252 (2009)
10. Ding, C.H., Li, T., Jordan, M.I.: Convex and Semi-nonnegative Matrix Factorizations. IEEE Transactions on Pattern Analysis and Machine Intelligence 99(1), 195–197 (2008)
11. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained K-means Clustering with Background Knowledge. In: Proceedings of the 18th International Conference on Machine Learning, Williamstown, MA, USA, pp. 577–584 (2001)
12. Dhillon, I., Mallela, S., Modha, D.S.: Information-Theoretic Co-clustering. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, pp. 89–98 (2003)

Rule Synthesizing from Multiple Related Databases*

Dan He[1], Xindong Wu[2,3], and Xingquan Zhu[4,5]

[1] Dept. of Computer Sci., Univ. of California Los Angeles, LA, CA, 90095, USA
[2] Dept. of Computer Sci., Univ. of Vermont, Burlington, VT, 05405, USA
[3] School of Computer Sci. & Info. Eng., Hefei Univ. of Tech., Hefei, 230009, China
[4] QCIS Center, Eng. & IT, Univ. of Technology, Sydney, NSW 2007, Australia
[5] FEDS Center, Chinese Academy of Sciences, Beijing, 100190, China
danhe@cs.ucla.edu, xwu@cems.uvm.edu, xqzhu@it.uts.edu.au

Abstract. In this paper, we study the problem of rule synthesizing from multiple related databases where items representing the databases may be different, and the databases may not be relevant, or similar to each other. We argue that, for such multi-related databases, simple rule synthesizing without a detailed understanding of the databases is not able to reveal meaningful patterns inside the data collections. Consequently, we propose a two-step clustering on the databases at both item and rule levels such that the databases in the final clusters contain both similar items and similar rules. A weighted rule synthesizing method is then applied on each such cluster to generate final rules. Experimental results demonstrate that the new rule synthesizing method is able to discover important rules which can not be synthesized by other methods.

Keywords: Association rule mining, rule synthesizing, multiple databases, clustering.

1 Introduction

Advances in data gathering, storage and distribution technologies introduce inevitable challenges of extracting meaningful patterns from related databases at distributed sites for unbiased pattern discovery and rapid decision making [11, 13]. Consider retail stores such as Walmart [10], which has more than 3800 stores in the US alone (and many more in other countries), each of which produces a huge number of transactions on a daily basis. The transactions are typically stored locally, which leads to a large number of related transaction databases. Notice that items sold in different countries/regions are rarely the same, such transaction databases are related to each other in terms of the data distributions and items representing the databases. Developing effective data

* This research has been supported by the U.S. National Science Foundation (NSF) under award CCF-0905337, the National Natural Science Foundation of China (NSFC) under grants 60828005, 60674109, the 973 Program of China under award 2009CB326203, and Australia Discovery Project DP1093762.

mining techniques to discover patterns from multiple related databases thus becomes crucial for these types of applications. Many methods exist for discovering patterns from databases sharing identical items, and existing solutions roughly fall into the following two categories: (1) **collective mining** – aggregating data from different databases to a centralized database for centralized processing [2, 5]; and (2) **distributed mining** – selecting important local rules from each individual database and synthesizing the local rules to form meaningful patterns over all related databases [7–9].

Compared to collective mining, distributed mining has the advantages of low transmission costs and low data privacy concerns [12, 16]. Therefore it attracts much attention recently. However, existing rule synthesizing methods for distributed mining commonly assumes that related databases are relevant, share similar data distributions, and have identical items. This is equivalent to the assumption that all stores have the same type of business with identical meta-data structures, which is hardly the case in practice. The irrelevance among related databases raises significant research issues to the rule synthesizing process, mainly from the following three challenges. (1) Databases with different items. Rule synthesizing is only able to synthesize rules containing items shared by all databases, but has to eliminate rules containing items unique in a few databases, even though the rules are locally significant. Therefore, in addition to discovering significant rules for all databases, solutions are needed to discover meaningful rules, which are locally significant and unique, for items which appear only in a few databases. (2) Databases with similar items but different rules. Although many databases may contain similar items, the rules underneath the data may vary significantly. Under such scenarios, the rule synthesizing process should produce meaningful rules which are locally significant and globally informative. (3) Customized rule synthesizing. For some applications, users may be either interested in rules related to some specific items, or a few items with the highest supports. This assumption is especially true when databases are collected from the Web, journals, books, etc. Under such a scenario, the rule synthesizing process should generate meaningful rules which make sense with respect to the users' queries or constraints.

In this paper, we report our recent work in addressing the above challenges, where the essential goal is to employ a two-step clustering based approach at the item and rule levels to ensure that important rules can be synthesized from related databases. For databases with different items (Challenge 1 above), clustering at the item level generates high-frequency items across all data collections. As a result, we only apply a rule synthesizing method on databases containing similar items. Such a clustering at the item level also helps to customize the rule synthesizing since the databases can be clustered according to the distance function built on the items specified by the users (Challenge 3). For databases sharing similar items but different rules (Challenge 2), the clusters generated from the item-level clustering are further clustered using a rule based distance function. So databases are clustered according to the high-frequency association rules they contain. Thus the final clusters contain not only similar high-frequency items but also similar high-frequency rules associated with these items.

The remainder of the paper is structured as follows. In Section 2 we summarize related work. In Section 3 we present the proposed two-step clustering based rule synthesizing framework, followed by experimental comparisons in Section 4. The concluding remarks are reported in Section 5.

2 Related Work

Existing research on association rule mining from related databases roughly falls into the following two categories: (1) **collective mining** – aggregating data from different sources to a centralized database for centralized processing [2, 5]; and (2) **distributed mining** – selecting important local rules from each individual source and synthesizing rules to form meaningful patterns over all related databases [7–9].

For collective pattern mining, aggregating data from related sources is the most straightforward method but is not practical due to concerns such as bandwidth limitations, data ownership, and data privacy [16]. Alternatively, many methods exist for association rules mining without data integration [18–20]. Among them, count distribution, data distribution, and candidate distribution are three basic mechanisms [15]. For example, in [17], the authors proposed a fast distributed association rule mining method, where pattern pruning considers both local supports and the polling results exchanged between sites. In their problem setting, a dedicated master site is used to control related databases to carry out pattern mining. The main theme of collective mining methods, from distributed association rule mining perspective, is to discover association rules which are globally significant from all related databases' point of view. The primary technical challenge is to minimize the communication cost between distributed sites in order to accelerate the pattern pruning process. For all above methods, the underlying databases must have identical items. In comparison, our research does not intend to find rules globally significant, but focuses on discovering rules from databases with different items.

Different from collective mining, where both related databases and the master site are involved to achieve a mining goal, in distributed mining, the primary focus is to combine mining results from distributed sites to synthesize new/significant rules [1, 21]. Under this framework, the related databases independently carry out the mining activities, and the results (rules) are aggregated to synthesize global interesting rules. Wu and Zhang [7, 8] proposed a weighting model to effectively synthesize local promising rules. After all promising rules are selected from different databases using normal association rule mining algorithms such as Aprior [4], a weight is then assigned to each rule, according to the number of databases that contain the rule. Then a weight for each database is computed according to their rules and the rule weight values. Using the weights, a global support for each rule is computed and is used to determine significant rules overall all data collections. However, the weighting model assumes the databases are relevant and share highly-similar items and rules so it can not handle irrelevant databases properly. Since the model doesn't do clustering, we call it *simple synthesizing algorithm*. The synthesized rules of this algorithm are always highly frequent globally.

Adhikari and Rao [9] proposed an algorithm to synthesize heavy association rules, which are the rules whose global supports are higher than a user given threshold. This criterion is the same as the measure defined in [7]. They also observed the cases that heavy association rules may not be shared by all databases. Therefore they defined a *highly-frequent rule* as the rule shared by at least $n \times \gamma_1$ databases and an *exceptional rule* as the rule shared by no more than $n \times \gamma_2$ databases, where n is the number of databases, and γ_1 and γ_2 are user defined thresholds. Then they synthesized the local rules and reported whether the rules are highly-frequent or exceptional. However, the algorithm still ignores the possible irrelevances among the databases. Since it synthesizes only heavy association rules, it is not able to report rules for items unique to certain groups of databases, and it is not able to report association rules for items the user inputs. What's more, the user needs to define both the highly-frequent and exceptional thresholds. As a conclusion, they addressed a different problem and their algorithm is not able to solve the challenges in Section 1.

3 Rule Synthesizing Based on Both Items and Rules

Intuitively, given a number of related databases, many rules may be frequent in only a few databases and a rule tends to be more significant if it is highly frequent in more databases. Accordingly, we define that a rule is **meaningful** if its support is higher than a given threshold t (we call t the *support threshold*) in a cluster of at least k similar databases(we call k the *cluster size threshold*), where t and k are given by users. Based on this definition, our two-step clustering method mainly aims to maximize the chance of capturing meaningful rules for synthesizing databases with dissimilar items and data distributions. It helps detect data similarities at both item and rule levels since databases may be dissimilar to each other at either level. Notice that if $k = 1$, the problem becomes simply mining normal frequent rules in individual databases, which collects significant rules from each database. If k equals the total number of databases, the simple synthesizing algorithm [7] works well since it always synthesizes rules globally, namely it considers all databases as in one cluster.

We define *r-RepItem (Representative Items)* as the r items with the highest supports in the list of items whose supports are greater than *minsupport*. The *r-RepItem* can also be r items of the user input if there is any. Then the **item based similarity** function is defined as

$$ISim(I_i, I_j) = \frac{|I_i \cap I_j|}{|I_i \cup I_j|}$$

where I_i, I_j are *r-RepItem* sets for data sources D_i, D_j. In the remaining of the paper, we will just use all items with supports higher than *minsupport* as the representative items.

3.1 Item Based Clustering Algorithm

We apply a maximal clique algorithm to generate appropriate clusters. We require for any two databases D_i, D_j in the same cluster, $sim(D_i, D_j) \geq \alpha$. The basic idea is to build a graph G=(V,E), where V=$D_1, D_2, ..., D_m$ is the m databases while $e_{i,j} \in E$ when $sim(I_i, I_j) \geq \alpha$, I_i is a representative item set of D_i, and α is the threshold of similarity. We enumerate all maximal cliques from the graph, using the popular algorithm from Bron and Kerbosch [6]. These maximal cliques can overlap with each other such that the clustering always exits for any threshold α. This is useful when the user input threshold is allowed. If we do not allow clusters to overlap, namely if we require $sim(I_i, I_j) < \alpha$ for D_i and D_j in different clusters, the clusters may not exist for some threshold. Each clique will be assigned one cluster label. Since the maximal cliques are always the same for a given graph, the clustering result never changes with the order of input databases. We call this algorithm **Maximal Clique Clustering Algorithm (MCCA)**.

Our purpose is to find a clustering where databases from the same cluster are similar to each other while databases from different clusters are different from each other. Therefore a key challenge is to design an evaluation function which has a unique polar-point, namely a unique minimum or maximum value, for a set of given databases and thus can be used to select the best clustering automatically by the algorithm shown in Figure 1. We define a distance function as follows.

$$Value(C^\alpha) = \sum_{I_i, I_j \in C^\alpha}^{i \neq j} (1 - ISim(I_i, I_j)) \tag{1}$$

$$dist(C_i, C_j) = \sum_{I_i \in C_i, I_j \in C_j} ((1 - ISim(I_i, I_j)) \tag{2}$$

$$Goodness(C, \alpha) = \Sigma_{i=1}^{n} Value(C_i^\alpha) \tag{3}$$

$$distance(C, \alpha) = Goodness(C, \alpha) - \sum_{i \neq j} (dist(C_i, C_j)) \tag{4}$$

Here C stands for clusters, C^α stands for the clusters when the similarity threshold is α. C_i^α stands for the ith cluster when the similarity threshold is α. $ISim$ stands for item based similarity, and $dist(C_i, C_j)$ is the distance between two clusters. The smaller the distance function $distance(C, \alpha)$ is, the better a clustering is. This function has at least one polar-point for a set of given databases since its value can never be infinite and its value decreases first and then increases. Therefore it can be used to select the best clustering automatically.

The distance function in Equation 4 has a unique polar-point such that the algorithm shown in Figure 1 can be applied directly to select the best clustering.

Lemma 1. *Distance function* $distance(C, \alpha)$=$Goodness(C, \alpha)$-$\sum_{i \neq j}(dist(C_i, C_j))$ *has a unique polar-point.*

```
Select Best Clustering Algorithm
Input: $D_1, D_2, ..., D_t$: databases; $\lambda$: the amount the threshold being updated each time
Output: set of clusters;
Given any similarity threshold $\alpha$, we can obtain a set of clusters by MCCA, $Cluster \leftarrow MCCA(D_1, D_2, ..., D_t, \alpha)$, for distance function $distance(Cluster, \alpha)$. We want to find the optimal $\alpha$ automatically.
1: Initialize $\alpha_1 \leftarrow 1$;
      value of distance function $x_1 \leftarrow distance(Cluster, \alpha_1)$;
2: Do
      Reduce $\alpha_1$ each time by $\lambda$.
      Record old $\alpha_1$, $x_1$ and new threshold $\alpha_2 \leftarrow \alpha_1 - \lambda$,
      new distance value $x_2 \leftarrow distance(Cluster, \alpha_2)$;
   While (the value of distance function decreases, namely $x2 \leq x1$)
3: If value of distance function starts to increase, namely $x2 \geq x1$
      Apply binary search on the range $[\alpha_2, \alpha_1]$ to find the $\alpha$ such that the corresponding
      $x \leftarrow distance(Cluster, \alpha)$ is the polar-point of the distance function.
4: output $cluster \leftarrow MCCA(D_1, D_2, ..., D_t, \alpha)$;
5: end all.
```

Fig. 1. Select Best Clustering Algorithm

The proof of Lemma 1 is omitted due to space restrictions. Since we allow overlapping among different clusters, a best clustering always exists.

Given that the distance function has a unique polar-point, we are able to find the polar-point automatically, using the algorithm shown in Figure 1. The algorithm initializes the similarity threshold as 1 and then keeps on decreasing the threshold by a small number λ until it locates a small region that contains the polar-point. Then it uses binary search to find the polar-point. It is straightforward to see the correctness of the algorithm. The distance function where $ISim(I_i, I_j)$ is used can be applied to the algorithm to select the best clustering. We call the algorithm using $ISim(I_i, I_j)$ *RuleBasedClustering* Algorithm. In the algorithm, we set $\lambda = maxSim/20$ where $maxSim$ is the maximum similarities among all databases, and the denominator is set to 20 empirically.

We observe that the maximal clique clustering algorithm may not be feasible to a large dense graph where the number of edges is close to $O(n^2)$ and n is the number of nodes. This is because the number of maximal cliques may increase exponentially with the number of edges. Based on this observation, we propose a **Greedy Clique Clustering Algorithm (GCCA)**. A *maximum* clique, which is of the largest size among all maximal cliques, is found at first, then all the nodes in the clique are removed from the graph. Then the next maximum clique is found in the remaining graph. The algorithm runs until there is no more node to remove. However, the greedy selection of the maximum clique may not generate the best clustering since once the nodes of a maximum clique are removed, the process cannot be reverted. If there are several maximum cliques, we select the one with the smallest value as defined in Formulae 1 and 5, respectively. Since the characteristics of clusters generated by **GCCA** are the same as those generated by **MCCA**, the new distance function can be applied to **MCCA** directly such that a best clustering at each step can be found automatically. One problem of **GCCA** is it may produce some small clusters whose sizes are smaller than the given cluster size threshold k since databases are removed at each step. Therefore the possibly important rules from these clusters may be missing. To address this problem, we take the rules from these clusters to be synthesized using the method in [7], which always synthesizes rules globally without clustering. This guarantees that **GCCA** captures more meaningful rules than the method in [7].

3.2 Rule Based Clustering Algorithm

Once the representative item based clustering is done, we obtain a set of clusters where databases inside the same cluster contain similar representative items. However, even though the databases in one cluster are relevant to each other, namely they contain similar representative items, they may still contain very different rules related to these items. These rules will not be generated by the synthesizing algorithm even though they are actually very meaningful.

To solve this problem, we need to apply step two of our clustering algorithm, namely rule based clustering, on each cluster obtained from the item based clustering (we call these clusters **Step 1 Clusters**). The clustering process is similar to the item based one, but with a different similarity function. We first need to select global representative items in each step 1 cluster. We can apply the weighting model on items for each step 1 cluster and select the first r items with the highest global weights. If the user inputs r items that they are interested in, these items will then be used instead of the r items with the highest global weights. Then for each such item, we mine association rules related to the item (namely rules contain the item) in the databases in each step 1 cluster. We then define a similarity function for these rules. Let D_i, D_j be two databases, S_i, S_j be the sets of association rules from D_i, D_j related to item d, respectively, and $|S|$ be the number of association rules in set S. We define the **rule based similarity** between D_i and D_j as

$$RSim(d, S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$$

$$Value(C^\alpha) = \Sigma_{S_i, S_j \in C^\alpha}^{i \neq j}(1 - RSim(d, S_i, S_j)) \tag{5}$$

$$dist(C_i, C_j) = \Sigma_{S_i \in C_i, S_j \in C_j}((1 - RSim(d, S_i, S_j)) \tag{6}$$

Here C stands for a cluster, and $RSim$ stands for rule based similarity. The distance function of rule based clustering will be the same as the one from item based clustering while using rule based *Value* and *dist* functions. The *RuleBasedClustering* algorithm is the same as the *ItemBasedClustering* algorithm with the only difference that $RSim(d, S_i, S_j)$ is used in the algorithm. Once the best clustering is selected, the weighting model will be applied to each subcluster to generate high-frequency rules related to each global representative item.

3.3 Weighting Model for Rule Synthesizing

After the two-step clustering, we have clusters of databases which share similar items and similar rules. We then apply the weighting model [7] as shown in Figure 2 to synthesize rules on each cluster. The model is shown to be effective

```
**RuleSynthesizing Algorithm**
**Input**: $S_1, S_2, \cdots, S_m$: rule sets; $minsupp$: threshold value;
**Output**: $X \rightarrow Y$: synthesized association rules;
1. let $S \leftarrow \{S_1, S_2, \cdots, S_m\}$;
2. for each rule $R$ in $S$ do
3.     let $Num(R) \leftarrow$ the number of data sources that
        contain rule $R$ in $S$;
4.     let $w_R \leftarrow \frac{Num(R)}{\sum_{R' \in S} Num(R')}$;
5. for $i = 1$ to $m$ do
6.     let $w_i \leftarrow \frac{\sum_{R_k \in S_i} Num(R_k) * w_{R_k}}{\sum_{j=1}^{m} \sum_{R_h \in S_j} Num(R_h) * w_{R_h}}$;
7. for each rule $X \rightarrow Y \in S$ do
8.     let $supp_w \leftarrow w_1 * supp_1 + w_2 * supp_2 + \cdots + w_m * supp_m$;
9. rank all rules in $S$ by their supports;
10. output the high-rank rules in $S$ whose supports are at least $minsupp$;
11. end all.
```

Fig. 2. Rule Synthesizing Algorithm [7]

```
**Two-Step Clustering Based Rule Synthesizing Algorithm**
**Input**: $D_1, D_2, ..., D_m$ databases, $minsupp$, threshold $\alpha$, the number of representative items $r$.
**Output**: $C_1, C_2, ..., C_k$ clusters, $X \rightarrow Y$: synthesized association rules for each cluster, $r$-$RepItem$ for each cluster.
1. generate $r$-$RepItem$ $I_1, I_2, ..., I_m$;
2. generate Step 1 clusters $[C_{11}, C_{12}, ..., C_{1h}] \leftarrow ItemBasedClustering(I_1, I_2, ..., I_m)$;
3. for $1 \leq i \leq h$ do
4.     generate $r$-$RepItem$ for Step 1 cluster $C_{1i}$.
       $I \leftarrow ItemSynthesizing(C_{1i}, minsupp)$;
5.     for item $d \in I$ do
6.         generate rule sets $S_1, S_2, ..., S_t$ for
           $D_1, D_2, ..., D_t \in C_{1i}$;
7.         generate Step 2 clusters $[C_{21}, C_{22}, ..., C_{2k}] \leftarrow RuleBasedClustering(S_1, S_2, ..., S_t)$;
8.         for $1 \leq j \leq k$ do
9.             synthesize rule set $S_{i,j} \leftarrow RuleSynthesizing(C_{2j}, minsupp)$;
10.            output item d, rule set $S_{i,j}$ and cluster $C_{2j}$;
11.        end for
12.    end for
13. end for
```

Fig. 3. Two-Step Clustering Based Rule Synthesizing Algorithm. *ItemBased Clustering*() and *RuleBasedClustering*() apply the MCCA or GCCA in SelectBest-ClusteringAlgorithm.

in synthesizing rules from multiple databases. After all promising rules are selected from different databases using normal association rule mining algorithms such as Apriori [4] , a weight is then assigned to each rule, according to the number of databases that contain the rule. Then a weight for each database is computed according to their rules and the rule weight values. Using the weights, a global support for each rule is computed, and is used to determine significant rules over all data collections. We show our two-step clustering based algorithm in pesudocode in Figure 3. The algorithm selects best clustering using the above representative item-based distance function and then synthesize the items to generate representative items for each cluster. Then it further selects the best clustering for each Step 1 cluster using the rule-based distance function. Finally it synthesizes the rules from databases in each final cluster. The functions *Item-Synthesizing* and *RuleSynthesizing* both use the weighting model on items and

database	#items	#transactions	size of transaction
Balance	23	625	5
BC	53	286	10
Nursery	32	12960	9

Fig. 4. Characteristics of databases Balance, Breast Cancer, Nursery

rules, respectively. Globally meaningful items and rules on each input cluster are then generated by these two functions.

4 Experimental Results

We implemented our two-step clustering based algorithm on a 2.8GHz P4 CPU with 512MB memory. To demonstrate the algorithm performance on real-world data, we applied it to several machine learning benchmark datasets from the UCI data repository [22]. In order to construct appropriate transaction databases, we consider each different value of each attribute including the class attribute as an item and each instance as a transaction. Therefore all the values in the same line are the items in the same transaction. We construct the transaction databases from three databases: Balance, BC (Breast Cancer) and Nursery. The characteristics of each database are summarized in Figure 4.

4.1 Experimental Results on Similar Databases

We first conducted the experiments on the transaction database transferred from the Balance dataset. We randomly split the transaction database into 10 databases of roughly equal sizes. Then we apply the two-step clustering based algorithm on the 10 databases. We set the *minsupport* as 0.02. We also set the cluster size threshold as 2, given that there are only 10 databases. Therefore the synthesized rules are meaningful if their supports are higher than 0.02 in a cluster of at least two databases. We apply **MCCA** as our clustering method. We compared the number of meaningful rules captured from each database by four methods: regular Apriori on each database, synthesizing without clustering [7], synthesizing with item-based clustering only and synthesizing with the two-step clustering based algorithm. For the Apriori algorithm, we simply select significant rules whose supports are greater than the *minsupport* from each database.These numbers illustrate our method filters out lots of rules significant in only individual databases. Our algorithm runs for a few seconds. We show the experimental results in the left graph of Figure 5. As we expected, the item-based clustering only algorithm and the two-step clustering based algorithm capture much more meaningful rules with high supports in at least two databases than the simple synthesizing method does. We can also observe that item-based only algorithm and two-step clustering based algorithm capture the same number of rules. This is because in each database the number of items is very small compared to the number of transactions (23 vs. 62×2^5). Once the databases are

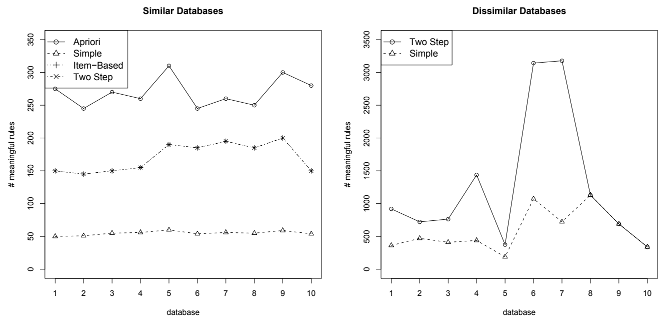


Fig. 5. Experiments on similar and dissimilar databases generated by splitting the Balance and BC databases into 10 databases of similar sizes, respectively. For similar databases, Representative Item support threshold = 0.1. Rule support threshold = 0.02. Cluster size threshold = 2. For dissimilar databases, Representative Item support threshold = 0.02. Rule support threshold = 0.04. Cluster size threshold = 2.

clustered based on item distance, the databases in the same cluster contain similar items. Since the item number is small, most transactions will contain similar items, which generates similar rules. Therefore these databases are similar on the rule level and the clusters remain unchanged after rule-based clustering.

4.2 Experimental Results on Dissimilar Databases

In order to illustrate the benefits of our algorithm, we conducted the second set of experiments on the transaction database transferred from the BC dataset. Again, we split the transaction database into 10 databases of roughly equal sizes. However, for the second half of the databases, we injected random transactions containing different items selected from a randomly generated itemset. The injected transactions are of different sizes. The number of injected transactions for each database is the same as that of the original database. Therefore, after the injection, the injected databases are highly possibly less similar to the unchanged databases while the unchanged databases are still similar to each other. The *minsupport* for item-based clustering and rule-based clustering are fixed as 0.02 and 0.04, respectively. The cluster size threshold is again fixed as 2. We compare the numbers of meaningful rules captured from each database by the simple synthesizing method [7] and our two step clustering based method. We use **GCCA** for clustering. Our two-step clustering based algorithm ran for around 1 minute. We show the experimental results in the right graph of Figure 5. The item-based clustering correctly clusters the unchanged databases (databases 1 to 5) which are more similar to each other. For the databases with injected transactions (databases 6 to 10), since the transactions are injected randomly, these injected databases are more likely to be different from each other but may still stand a chance to be similar to other databases (databases 6, 7, e.g.). For the clusters whose size is less than two, the two-step clustering based algorithm

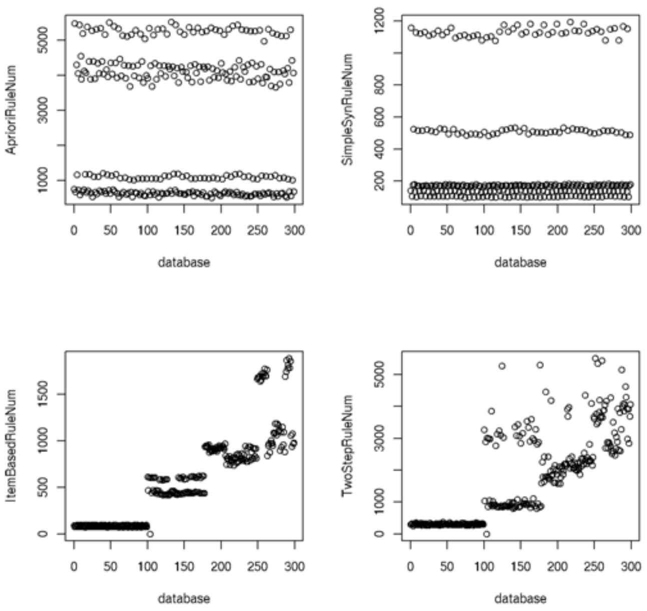


Fig. 6. The original database Nursery was first transferred to a transaction database and then was split into 300 databases of equal size. Representative Item support threshold = 0.1. Rule support threshold = 0.08.

simply takes the rules synthesized by the simple synthesizing method. That's why the number of synthesized rules for some databases such as databases 8, 9, 10, are the same for both methods. The two-step clustering based algorithm using **GCCA** always captures no less meaningful rules than the simple synthesizing method does. Those newly captured rules are all highly frequent in at least two similar databases. If we use **MCCA**, an optimal clustering will be found and we can capture more meaningful rules from those databases which are in clusters of size less than two by **GCCA**. Due to space restrictions, a comparison between **MCCA** and **GCCA** is omitted.

4.3 Scalability Assessment

The last experiment is to show that our algorithm is able to scale up to a large number of databases. We conduct the experiment on the Nursery database. This database is much larger than the previous two databases. We randomly split the database into 300 databases of equal size. Again, we inject random transactions to half of them, however, unlike the previous experiment, we inject random transactions alternatively to the database, namely random transactions are injected for databases with each odd index in the range of [1, 300]. The two-step clustering algorithm ran for 104 minutes. We show the numbers of meaningful rules captured from each database by the four methods in Figure 6. To illustrate how the databases are clustered, we didn't set the cluster size threshold. Since the number of databases is relatively big, for illustration purpose, we use **GCCA** for clustering. As we can see in the first plot for the Apriori algorithm, since random transactions were injected to all databases with an odd index, these databases contain more rules, while the remaining unchanged databases contain less rules. A similar rule number distribution can be observed for the simple synthesizing method where the number of meaningful rules captured in each database is much less than that by the Apriori algorithm. For Item-based only synthesizing and two-step clustering based synthesizing, we re-arrange the indices of the databases such that the databases in the same cluster are indexed close to each other. Clearly we can see the databases are clustered into a few clusters where the distributions of the numbers of meaningful rules for the databases in each cluster are similar. Again, the number of meaningful rules captured by the two-step clustering based algorithm is more than the item-based only algorithm. One more observation is although we didn't fix the cluster size threshold, it is easy to see that there are many clusters of sizes more than ten after the item-based only clustering and the two-step clustering. The numbers of highly frequent rules from them are many more than those from the simple synthesizing method. It indicates if we set the cluster size threshold as 10, our method is able to capture much more meaningful rules.

5 Conclusions

In this paper, we proposed a general rule synthesizing framework for association rule mining from multiple related databases. We argued that due to realities that (1) multi-related data collections may have different items, (2) local patterns in the related databases may be largely different, and (3) users may require a rule synthesizing process to be customized to some specific items, existing rule synthesizing methods are ineffective in solving all these challenges. Alternatively, we proposed a two-step clustering based rule synthesizing framework, which clusters the data at both item and rule levels, to synthesize association rules from multiple related databases. In our definition, a synthesized rule is meaningful if its support is frequent in a cluster of related databases. A synthesizing algorithm is then applied on the final clusters to find significant rules for representative items of the clusters. Experimental results and comparisons indicated that the proposed two-step clustering based rule synthesizing method is able to capture meaningful rules that are otherwise incapable of being synthesized by other methods.

References

1. Zhang, S., Zhang, C., Wu, X.: Knowledge Discovery in Multiple Databases. Springer, Heidelberg (2004)
2. Skillicorn, D., Wang, Y.: Parallel and Sequential Algorithms for Data Mining Using Inductive Logic. Knowledge and Info. Syst. 4, 405–421 (2001)

3. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Databases. In: Proc. of SIGMOD, pp. 207–216 (1993)
4. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Proc. Very Large Database Conf. (1994)
5. Parthasarathy, S., Zaki, M., Ogihara, M., Li, W.: Parallel Data Mining for Association Rules on Shared-Memory Systems. In: Knowledge and Information Systems, March 2001, vol. 1, pp. 1–29 (2001)
6. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. Communications of the ACM 16, 575–577 (1973)
7. Wu, X., Zhang, S.: Synthesizing High-Frequency Rules from Different databases. IEEE Trans. Knowledge and Data Eng. 15(2), 353–367 (2003)
8. Wu, X., Zhang, C., Zhang, S.: Database classification for multi-database mining. Information Systems 30, 71–88 (2005)
9. Adhikari, A., Rao, P.R.: Synthesizing Heavy Association Rules from Different Real Data Sources. Pattern Recognition Let. 29(1), 59–71 (2008)
10. http://en.wikipedia.org/wiki/Wal-Mart
11. Manjhi, A., Shkapenyuk, V., Dhamdhere, K., Olston, C.: Finding (recently) frequent items in distributed data streams. In: Proc. of ICDE (2005)
12. Vaidya, J., Clifton, C.: Privacy Preserving Association Rule Mining in Vertically Partitioned Data. In: Proc. of KDD (2002)
13. Tan, P., Jin, R.: Ordering Patterns by Combining Options from Multiple Sources. In: Proc. of KDD (2004)
14. http://www.cs.loyola.edu/~cgiannel/assoc_gen.html
15. Kargupta, H., Park, B.H., Hershberger, D., Johnson, E.: Collective data mining: A new perspective toward distributed data mining. In: Advances in Distributed and Parallel Knowledge Discovery. MIT/AAAI Press, Cambridge (1999)
16. Kantarcioglu, M., Clifton, C.: Privacy-preserving distributed mining of association rules on horizontally partitioned data. In: ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'02) (June 2002)
17. Cheung, D., Ng, V., Fu, A., Fu, Y.: Efficient Mining of Association Rules in Distributed Databases. IEEE Transactions on Knowledge and Data Engineering (1996)
18. Schuster, A., Wolff, R.: Communication-efficient distributed mining of association rules, Data Mining and Knowledge Discovery. Data Mining and Knowledge Discovery 8(2), 171–196 (2004)
19. Skillicorn, D.B.: Parallel frequent set counting. Distributed and Parallel Databases 28(5), 815–825 (2002)
20. Otey, M., Veloso, A., Wang, C., Parthasarathy, S., Meira, W.: Mining frequent itemsets in distributed and dynamic databases. In: Proc. of ICDM Conference (2003)
21. Su, K., Huang, H., Wu, X., Zhang, S.: A logical framework for identifying quality knowldge from different data sources. Decision Support Systems 42(3) (2006)
22. Asuncion, A., Newman, D.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine (2007), http://www.ics.uci.edu/~mlearn/MLRepository.html

Fast Orthogonal Nonnegative Matrix Tri-Factorization for Simultaneous Clustering

Zhao Li[1], Xindong Wu[1,2], and Zhenyu Lu[1]

[1] Department of Computer Science, University of Vermont, Unite States
[2] School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, PR China
{zhaoli,xwu,zlu}@cems.uvm.edu

Abstract. Orthogonal Nonnegative Matrix Tri-Factorization (ONMTF), a dimension reduction method using three small matrices to approximate an input data matrix, clusters the rows and columns of an input data matrix simultaneously. However, ONMTF is computationally expensive due to an intensive computation of the Lagrangian multipliers for the orthogonal constraints. In this paper, we introduce Fast Orthogonal Nonnegative Matrix Tri-Factorization (FONT), which uses approximate constants instead of computing the Lagrangian multipliers. As a result, FONT reduces the computational complexity significantly. Experiments on document datasets show that FONT outperforms ONMTF in terms of clustering quality and running time. Moreover, FONT is further accelerated by incorporating Alternating Least Squares, and can be much faster than ONMTF.

Keywords: Nonnegative Matrix Factorization, Orthogonality, Alterative Least Square.

1 Introduction

Dimension reduction is a useful method for analyzing data of high dimensions so that further computational methods can be applied. Traditional methods, such as principal component analysis (PCA) and independent component analysis (ICA) are typically used to reduce the number of variables and detect the relationship among variables. However, these methods cannot guarantee nonnegativity, and are hard to model and interpret the underlying data. Nonnegative matrix factorization (NMF) [7,8], using two lower-rank nonnegative matrices $W \in \mathbb{R}^{m \times k}$ and $H \in \mathbb{R}^{k \times n}$ to approximate the original data $V \in \mathbb{R}^{m \times n}$ $(k \ll min(m, n))$, has gained its popularity in many real applications, such as face recognition, text mining, signal processing, etc [1].

Take documents in the vector space model for instance. The documents are encoded as a term-by-document matrix V with nonnegative elements, and each column of V represents a document and each row a term. NMF produces k basic topics as the columns of the factor W and the coefficient matrix H. Observed from H, it is easy to derive how each document is fractionally constructed by the resulting k basic topics. Also, the factor H is regarded as a cluster indicator matrix for document clustering, each row of which suggests which documents are included in a certain topic. Similarly, the factor W can be treated as a cluster indicator matrix for word clustering. Traditional clustering algorithms, taking k-means for instance, require the product between the row vectors or column vectors to be 0 that only one value exists in each row of H; thus each data point only belongs to one cluster, which leads to a hard clustering. It was proved that orthogonal nonnegative matrix factorization is equivalent to k-means clustering [4]. Compared to rigorous orthogonality of k-means, relaxed orthogonality means each data point could belong to more than one cluster, which can improve clustering quality [5,10]. Simultaneous clustering refers to clustering of the rows and columns of a matrix at the same time. The major property of simultaneous clustering is that it adaptively performs feature selection as well as clustering, which improves the performance for both of them [2,3,6,12]. Some applications such as clustering words and documents simultaneously for an input term-by-document matrix, binary data, and system log messages were implemented [9]. For this purpose, Orthogonal Nonnegative Matrix Tri-Factorization (ONMTF) was proposed [5]. It produces two nonnegative indictor matrices W and H, and another nonnegative matrix S such that $V \approx WSH$. Orthogonality constraints were imposed on W and H to achieve relaxed orthogonality. However, in their methods, to achieve relaxed orthogonality, Lagrangian multipliers have to be determined for the Lagrangian function of ONMTF. Solving the Lagrangian multipliers accounts for an intensive computation of update rules for the factors, especially the factor W whose size is larger than other factors. In this paper, we introduce Fast Orthogonal Nonnegative Matrix Tri-Factorization (FONT), whose computational complexity is decreased significantly by setting the Lagrangian multipliers as approximate constants. In addition, FONT is further accelerated by using Alternating Least Squares [11], which leads to a fast convergence.

The rest of the paper is organized as follows. In Section 2, related work is reviewed, including NMF and ONMTF. Section 3 introduces our methods in detail, followed by the experiments and evaluations in Section 4. Finally, conclusions are described in Section 5.

2 Related Work

Given a data matrix $V = [v_1, v_2, ..., v_n] \in \mathbb{R}^{m \times n}$, each column of which represents a sample and each row a feature. NMF aims to find two nonnegative matrices $W \in \mathbb{R}^{m \times k}, H \in \mathbb{R}^{k \times n}$, such that $V \approx WH$, where $k \ll min(m, n)$. There is no guarantee that an exact nonnegative factorization exists. Iterative methods become necessary to find an approximate solution to NMF which is a nonlinear optimization problem with inequality constraints. To find an approximate factorization of NMF, an objective function has to be defined by using some measurements of distance. A widely used distance measurement is the Euclidean distance which is defined as:

$$\min_{W,H} \|V - WH\|^2 \quad s.t. \quad W \geq 0, \quad H \geq 0 \tag{1}$$

where the $\|\cdot\|$ is Frobenius norm. To find a solution to this optimization problem, the multiplicative update rules were first investigated in [8] as follows:

$$W := W * (VH^T)/(WHH^T) \tag{2}$$

$$H := H * (W^T V)/(W^T WH) \tag{3}$$

where * and / denote elementwise multiplication and division, respectively. ONMTF was conducted for the application of clustering words and documents simultaneously by imposing additional constraints on W and/or H. The objective function for ONMTF can be symbolically written as:

$$F = \min_{W,S,H \geq 0} \|V - WSH\|^2 \quad s.t. \quad HH^T = D, \quad W^T W = D \tag{4}$$

where D is a diagonal matrix. By introducing the Lagrangian multipliers the Lagrange L is:

$$L = \|V - WSH\|^2 + Tr[\lambda_w(W^T W - D)] + Tr[\lambda_h(HH^T - D)] \tag{5}$$

The multiplicative update rules for (8) can be computed as follows:

$$W = W * (VH^T S^T)/(W(HH^T + \lambda_w)) \tag{6}$$

$$S = S * (W^T WH^T)/(W^T SHH^T) \tag{7}$$

$$H = H * (S^T W^T V)/((W^T W + \lambda_h)H) \tag{8}$$

By solving the minimum $W^{(t+1)}$ and $H^{(t+1)}$, respectively [5]. λ_w and λ_h can be approximately computed as follows:

$$\lambda_w = D^{-1} W^T VH^T S^T - HH^T \tag{9}$$

$$\lambda_h = D^{-1} S^T W^T VH^T H - W^T W \tag{10}$$

Substituting λ_w and λ_h in (7) and (8) respectively, we obtain following update rules:

$$W = W * (VH^T S^T)/(WW^T VH^T S^T) \tag{11}$$

$$H = H * (S^T W^T V)/(S^T W^T VH^T H) \tag{12}$$

Based on matrix multiplication, the computational complexity of NMF based on the Euclidean distance metric at each iteration is $O(mnk)$, and that of ONMTF is $O(m^2 n)$. The computation of the Lagrangian multipliers accounts for an intensive computation. It becomes worse when m increases, which represents the number of words in a vector space model.

3 Fast Orthogonal Nonnegative Matrix Tri-Factorization

It was proved that the Lagrange L is monotonically non-increasing under the above update rules by assuming $W^TW + \lambda_w \geq 0$ and $HH^T + \lambda_h \geq 0$ [5]. We note that λ_w and λ_h are approximately computed under these assumptions, and from (9) and (10) we can see that λ_w and λ_h are symmetric matrices of size $k * k$. Since achieving relaxed orthogonality is the purpose of orthogonal matrix factorization in this paper, and computing the lagrangian multipliers accounts for an intensive computation, we would use constants for λ_w and λ_h for decreasing computational complexity. By normalizing each column vector of W and each row vector of H to unitary Euclidean length at each iteration, λ_w and λ_h can be approximately denoted by minus identity matrix ($\lambda_w = \lambda_h = -I$). Thus, we introduce our method *Fast Orthogonal Nonnegative Matrix Tri-Factorization* (FONT).

3.1 FONT

The Lagrange L is rewritten as:

$$L = \min_{W,S,H \geq 0} (\|V - WSH\|^2 + Tr(I - W^TW) + Tr(I - HH^T)) \qquad (13)$$

where I is the identity matrix. Noting $\|V - WSH\|^2 = Tr(VV^T) - 2Tr(WSHV^T) + Tr(WSHH^TS^TW^T)$, the gradient of L with respect to W and H are:

$$\partial L/\partial W = -2VH^TS^T - 2W + 2WSHH^TS^T \qquad (14)$$

$$\partial L/\partial H = -2S^TW^TV - 2H + 2S^TW^TWSH \qquad (15)$$

By using the Karush-Kuhn-Tucker conditions the update rules for W and H can be inferred as follows:

$$W = W * (VH^TS^T + W)/(WSHH^TS^T) \qquad (16)$$

$$H = H * (S^TW^TV + H)/(S^TW^TWSH) \qquad (17)$$

Because of no orthogonality constraint on S, the update rule for S, in both FONT and ONMTF, is the same. The computational complexity of FONT, at each iteration, is $O(mnk)$, far less than $O(m^2n)$ because $k \ll min(m, n)$.
Now we give the convergence of this algorithm by using the following theorem (we use H as an example here, and the case for W can be conducted similarly):

Theorem 1. *The Lagrange L in (13) is non-increasing under the update rule in (17).*

To prove this theorem, we use the auxiliary function approach [8]. $G(h, h')$ is an auxiliary function for $F(h)$ if the conditions $G(w, w') \geq F(w)$ and $G(w, w) = F(w)$ are satisfied. If G is an auxiliary function, then F is nonincreasing under the updating rule $w^{t+1} = arg\min_w G(w, w^t)$. Because $F(w^{t+1}) \leq G(w^{t+1}, w^t) \leq G(w^t, w^t) = F(w^t)$ [8]. So it is crucial to find an auxiliary function. Now we show that

$$G(h, h^t_{ij}) = L_{ij}(h^t_{ij}) + L'_{ij}(h^t_{ij})(h - h^t_{ij}) + (h - h^t_{ij})^2 * (S^T W^T W S H)_{ij} / h^t_{ij} \quad (18)$$

is an auxiliary function for L.

Proof. Apparently, $G(h, h) = L_{ij}(h)$, so we just need to prove that $G(h, h^t_{ij}) \geq L_{ij}(h)$. We expand $L_{ij}(h)$ using Taylor series.

$$L_{ij}(h) = L_{ij}(h^t_{ij}) + L'_{ij}(h^t_{ij})(h - h^t_{ij}) + [(S^T W^T W S)_{ii} - 1](h - h^t_{ij})^2 \quad (19)$$

Meanwhile,

$$\begin{aligned}(S^T W^T W S H)_{ij} &= \textstyle\sum_{p=1}^k (S^T W^T W S)_{ip} H_{pj} \\ &\geq (S^T W^T W S)_{ii} H_{ij} > ((S^T W^T W S)_{ii} - 1) h^t_{ij}\end{aligned} \quad (20)$$

Thus we have $G(h, h^t_{ij}) \geq L_{ij}(h)$. Theorem 1 then follows that the Lagrangian L is nonincreasing.

3.2 FONT + ALS

However, we still note that the factor W accounts for a larger computation than the other two factors, thus we consider to compute W by using Alternating Least Squares (ALS). ALS is very fast by exploiting the fact that, while the optimization problem of (1) is not convex in both W and H, it is convex in either W or H. Thus, given one matrix, the other matrix can be found with a simple least squares computation. W and H are computed by equations $W^T W H = W^T V$ and $H H^T W^T = H A^T$, respectively. Reviewing (4) for W, F can be rewritten as:

$$(W^T W + S H H^T S^T) W^T = W^T + S H V^T \quad (21)$$

Then an approximate optimal solution to W is obtained by using ALS. To maintain nonnegativity, all negative values in W should be replaced by zero.

4 Experiments

5 document databases from the CLUTO toolkit (`http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download`) were used to evaluate our algorithms. They are summarized in Table 1. Considering the large memory requirement for the matrix computation, we implemented all algorithms in Matlab R2007b on a 7.1 teraflop computing cluster which is an IBM e1350 with an 1 4-way (Intel Xeon MP 3.66GHz) shared memory machine with 32GB. The memory was requested between 1G to 15G for different datasets. 15G memory was required for the ONMTF algorithm to run on 2 largest datasets *la12* and *class*.

Table 1. Summary of Datasets

Dataset	Source	# Classes	# Documents	# Words
classic	CACM/CISI/Cranfield/Medline	4	7094	41681
reviews	San Jose Mercury(TREC)	5	4069	18483
klb	WebACE	6	2340	21839
la12	LA Times(TREC)	6	6279	31472
ohscal	OHSUMED-233445	10	11162	11465

4.1 Evaluation Metrics

We also use purity and entropy to evaluate the clustering performance [5]. Purity gives the average ratio of a dominating class in each cluster to the cluster size and is defined as:

$$P(k_j) = \frac{1}{k_j} max(h(c_j, k_j)) \tag{22}$$

where $h(c, k)$ is the number of documents from class c assigned to cluster k. The larger the values of purity, the better the clustering result is.
The entropy of each cluster j is calculated using the $E_j = \sum_i p_{ij} log(p_{ij})$, where the sum is taken over all classes. The total entropy for a set of clusters is computed as the sum of entropies of each cluster weighted by the size of that cluster:

$$E_C = \sum_{j=1}^{m} (\frac{N_j}{N} \times E_j) \tag{23}$$

where N_j is the size of cluster j, and N is the total number of data points. Entropy indicates how homogeneous a cluster is. The higher the homogeneity of a cluster, the lower the entropy is, and vice versa.

4.2 Performance Comparisons

In contrast to document clustering, there is no prior label information for word clustering. Thus, we adopt the class conditional word distribution that was used in [5]. Each word belongs to a document class in which the word has the highest frequency of occurring in that class. All algorithms (FONT$_{ALS}$ stands for FONT combined with ALS) were performed by using the stopping criterion $1 - F^{t+1}/F^t \leq 0.01$, where F is $\|V - WSH\|^2$ and calculated by every 100 iterations. The comparison for both word and document clustering are shown in Table 2 and Table 3 respectively. All results were obtained by averaging 10 independent trails.

We observe that the FONT algorithms (including FONT$_{ALS}$) achieve better purity than ONMTF for both words and documents clustering. It also shows that FONT obtains lower entropy for word clustering than ONMTF. But for document clustering, the clusters obtained by ONMTF are more homogenous than FONT and FONT$_{ALS}$. Meanwhile, in Table 4, it is shown that FONT and FONT$_{ALS}$ are significantly faster than ONMTF. In particular, the running time

Table 2. Comparison of Word Clustering

Dataset	Purity			Entropy		
	ONMTF	FONT	FONT$_{ALS}$	ONMTF	FONT	FONT$_{ALS}$
classic	0.5077	0.5153	0.5577	0.6956	0.6881	0.6309
reviews	0.5905	0.6298	0.6001	0.6850	0.6656	0.7003
klb	0.7258	0.7356	0.7335	0.4546	0.4486	0.4478
la12	0.4612	0.4823	0.4721	0.7693	0.7569	0.7794
ohscal	0.3740	0.4056	0.2991	0.7601	0.7297	0.8331

Table 3. Comparison of Document Clustering

Dataset	Purity			Entropy		
	ONMTF	FONT	FONT$_{ALS}$	ONMTF	FONT	FONT$_{ALS}$
classic	0.5484	0.5758	0.6072	0.6246	0.6359	0.6661
reviews	0.7312	0.7635	0.7540	0.7775	0.8097	0.8126
klb	0.8021	0.8095	0.8118	0.8317	0.8389	0.8366
la12	0.4978	0.5176	0.5379	0.5665	0.5883	0.6063
ohscal	0.3581	0.3983	0.3616	0.4305	0.4682	0.4340

of FONT$_{ALS}$ is 12.16 seconds on the largest dataset *classic*, compared to 36674 seconds ONMTF used and 1574.2 seconds NMTF used, which indicates that the FONT algorithms are effective in terms of clustering quality and running time.

Table 4. Comparison of Running Time (s)

Dataset	ONMTF	FONT	FONT$_{ALS}$
classic	3.6674e+4	1.5985e+3	**12.16**
reviews	2.0048e+4	370.36	**25.12**
klb	1.0852e+4	275.94	**14.07**
la12	4.5239e+4	1.0496e+3	**41.45**
ohscal	1.0051e+4	767.86	**37.29**

5 Conclusions

The Orthogonal Nonnegative Matrix Tri-Factorization algorithm needs a large computation to achieve relaxed orthogonality, which makes it infeasible for clustering large datasets in terms of computational complexity and a large requirement of memory. In our research, to achieve relaxed orthogonality, we have introduced our method Fast Nonnegative Matrix Tri-Factorization (FONT). By using unitary matrix to estimate the Lagrangian multipliers, the computational complexity is reduced and clustering quality is improved as well. Meanwhile, by using Alternating Least Squares, FONT is further accelerated, which leads to a significant decrease of running time.

Acknowledgements

This research is supported by the US National Science Foundation (NSF) under grant CCF-0905337 and EPS-0701410, the National Natural Science Foundation of China (NSFC) under award 60828005, and the 973 Program of China under award 2009CB326203.

References

1. Berry, M.W., Browne, M., Langville, A.N., Pauca, P.V., Plemmons, R.J.: Algorithms and applications for approximate nonnegative matrix factorization. Computational Statistics & Data Analysis 52(1), 155–173 (2007)
2. Boley, D.: Principal direction divisive partitioning. Data Min. Knowl. Discov. 2(4), 325–344 (1998)
3. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 89–98. ACM, New York (2003)
4. Ding, C., He, X., Simon, H.D.: On the equivalence of nonnegative matrix factorization and spectral clustering. In: SIAM Data Mining Conference (2005)
5. Ding, C., Li, T., Peng, W., Park, H.: Orthogonal nonnegative matrix trifactorizations for clustering. In: KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 126–135. ACM, New York (2006)
6. Koyutürk, M., Grama, A., Ramakrishnan, N.: Nonorthogonal decomposition of binary matrices for bounded-error data compression and analysis. ACM Trans. Math. Softw. 32(1), 33–69 (2006)
7. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature 401(6755), 788–791 (1999)
8. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Neural Information Processing Systems, pp. 556–562 (2001)
9. Li, T., Ding, C.: The relationships among various nonnegative matrix factorization methods for clustering. In: ICDM '06: Proceedings of the Sixth International Conference on Data Mining, Washington, DC, USA, pp. 362–371. IEEE Computer Society, Los Alamitos (2006)
10. Li, Z., Wu, X., Peng, H.: Nonnegative matrix factorization on orthogonal subspace. Pattern Recognition Letters (to appear, 2010)
11. Paatero, P., Tapper, U.: Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. Environmetrics 5(2), 111–126 (1994)
12. Zhang, Z., Li, T., Ding, C., Zhang, X.: Binary matrix factorization with applications. In: ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining, Washington, DC, USA, pp. 391–400. IEEE Computer Society, Los Alamitos (2007)

Hierarchical Web-Page Clustering via In-Page and Cross-Page Link Structures

Cindy Xide Lin[1], Yintao Yu[1], Jiawei Han[1], and Bing Liu[2]

[1] University of Illinois at Urbana-Champaign
{xidelin2,yintao}@uiuc.edu, hanj@cs.uiuc.edu
[2] University of Illinois at Chicago
liub@cs.uic.edu

Abstract. Despite of the wide diversity of web-pages, web-pages residing in a particular organization, in most cases, are organized with semantically hierarchic structures. For example, the website of a computer science department contains pages about its people, courses and research, among which pages of people are categorized into faculty, staff and students, and pages of research diversify into different areas. Uncovering such hierarchic structures could supply users a convenient way of comprehensive navigation and accelerate other web mining tasks. In this study, we extract a similarity matrix among pages via in-page and cross-page link structures, based on which a density-based clustering algorithm is developed, which hierarchically groups densely linked webpages into semantic clusters. Our experiments show that this method is efficient and effective, and sheds light on mining and exploring web structures.

1 Introduction

Web page clustering has been studied extensively in the literature as a means to group pages into homogeneous topic clusters. However, much of the existing study [1] [7] [18] [9] is based on any arbitrary set of pages, *e.g.*, pages from multiple websites. Limited work has been done on clustering pages from a specific website of an organization. Despite of the wide diversity of webpages, webpages residing in a particular organization, in most cases, have some semantically hierarchic structures. For example, the website of a computer science department may contain a large set of pages about its people, courses, news and research, among which pages of people can be categorized into the ones of faculty, staff and students, and pages of research may diversify into different areas. Uncovering such hierarchic structures could supply users a convenient way of comprehensive navigation, accelerate other searching and mining tasks, and enables us to provide value-added services.

This is, however, a challenging task due to the semantic and structural heterogeneity of the webpages. Nevertheless, one can observe that the information in a site is usually organized according to certain logical relationships, *e.g.*, related items are often organized *together* in a way that is easier for users to find relevant information items. For example, in a university department, there is usually a page about its faculty, a page about its courses, *etc.*. Exploring such site organizational information, *i.e.*, *information item togetherness*, will help us cluster the items of the same type.

From an implementation point of view, such togetherness is typically manifested in HTML code through two means: *in-page structures* and *cross-page hyper-links*. Information items of the same type are usually coded as sibling nodes of the same parent in the HTML tag tree (*i.e.*, *DOM tree*), and links that represent similar items often reside together in a page as siblings, forming *parallel links* of a page. Such page structure and parallel links provide an important clue in the design of similarity functions for meaningful clustering.

Based on this idea, we develop a novel method, HSClus, for hierarchical site clustering of webpages in order to discover the inherent semantic structure of an organization's website. Our major contributions include:

1. Deriving from *DOM* trees, a novel concept called *parallel links* is proposed, based on which a new similarity function between pages is developed.
2. A new clustering algorithm called HSClus is designed to group densely linked webpages into semantic clusters and identify their hierarchical relationships.

Our experiments show that HSClus is efficient and effective at uncovering webpage structures at some organization's website, which sets a foundation for further mining and exploring web semantic structures.

2 Related Work

Spectral partitioning [5] is a group of one-level network clustering algorithms, which targets to cutting a graph into a set of sub-graphs, *i.e.*, clusters, with an object function that minimizes the number of cross-cluster edges and maximizes the number of in-cluster edges. Because its solution relies on the calculation of eigen values, the time complexity is square to the number of edges. Agglomerative hierarchical clustering [11] [3] treats each data point as a singleton cluster, and then successively merges clusters until all points have been merged into a single remaining cluster. However, these methods are sensitive to outliers. DOM tree structures have been widely used for webpage segmentation and partitioning. As the correspondence of in-page parallel links in this paper, [14] enhances web page classification by utilizing labels and contents information from sibling pages. Web patterns [10] are formalized descriptions of common features of objects on web pages. Each page is presented by a vector of pattern weights, which record the extent of importance of the pattern for the web page. Based on pattern vectors, similarity between pages is defined. To automatically extract main classes of pages offered by a website, [4] compares structures of DOM trees. In order to improve search results via text contents [1] uses the path length and [7] uses weighted path between two pages to adjust clusters. [19] combines out-links, in-links and terms in page contents to improve the clustering quality on web search results. [2] finds dense units by density-based algorithms, and then merges units by agglomerative hierarchical clustering.

3 Similarity Matrix

This section takes a set of pages $P = \{p_0, \ldots, p_{n-1}\}$ at an organization's website as *input objects*, and outputs a *similarity matrix* among pages in P for the clustering algorithm introduced in latter sections.

Web pages contain abundant information about link structures that can help discovering web clusters, which is mainly in two categories: *cross-page link-structures* and *in-page link-structures*. The former one refers to the link graph among webpages, while the latter one refers to the organization of links inside an individual page. If we regard cross-page link-structures as web structures at the macro-level, then in-page link-structures are the one at the micro-level. Combining macro- and micro-levels of web structures will gain great power for link-based web page clustering.

3.1 Cross-Page Link-Structures

Co-citation [15] and bibliography-coupling [8] are two popular measures in the analysis of link graph. Concretely, for pages p_i and p_j, their co-citation $C(i,j)$ and bibliography-coupling $B(i,j)$ are defined as the frequencies of common in-links and out-links, respectively, saying $C(i,j) = \sum_k E(i,k)E(j,k)$ and $B(i,j) = \sum_k E(k,i)E(k,j)$, where $E(i,j) = 1$ if there is a hyper-link in p_i pointing to p_j and otherwise $E(i,j) = 0$. We use *Cosine* function to calculate the similarity $Sim_{CB}(i,j)$ gained from $C(i,j)$ and $B(i,j)$ for pages p_i and p_j [1]:

$$Sim_{CB}(i,j) = \frac{C(i,j)}{\sqrt{C(i,i) \cdot C(j,j)}} + \frac{B(i,j)}{\sqrt{B(i,i) \cdot B(j,j)}} \tag{1}$$

3.2 In-Page Link-Structures

The DOM (Document Object Model) is a platform- and language-independent standard object model for representing HTML or XML documents. Building DOM trees from input web pages is a necessary step for many data extraction algorithms. Furthermore, nodes in DOM trees are written in the form of tags, indicating the structure in a web page and a way of hierarchically arranging text-based contents. Formally, we use $DOM(i)$ to denote the DOM tree extracted from the source code of a particular web page p_i with trivial HTML tags removed. For a tree node μ in $DOM(i)$, the sub-tree rooted at μ is denoted by $DOM_\mu(i)$.

In this sub-section, we will introduce *Parallel Link* as a novel concept derived from DOM trees. Note parallel links are independently extracted from each page of the targeting website, which reasonably assumes the homogeneity of the layout and the contents inside one particular page, *e.g.*, the homepage of a laboratory may list hyper-links to its professors together and the link of each professor is followed by the professor's name and then by the email. Here the consecutive positions of these hyper-links and the homogeneous organization of each professor's information are good examples of in-page link-structures, which give strong

[1] Many other functions analyzed in [16] may also be good choices.

hints of the semantic meaning of these professors' pages. It is necessary to understand that it does not make any assumptions about the homogeneity of pages among the whole website.

Concretely, for sibling sub-trees[2] $DOM_{\mu_1}(i)$, $DOM_{\mu_2}(i)$, $\cdots$, $DOM_{\mu_k}(i)$, they become a group of *Parallel Sub-Trees* if $DOM_{\mu_s}(i)$ and $DOM_{\mu_t}(i)$ for any $s, t \in 1..k$ are exactly the same (including the tree structures and the HTML tags). Tree nodes ν_1, ν_2, $\cdots$, ν_k form a group of *Parallel Nodes* if they locate in the same position of $DOM_{\mu_1}(i)$, $DOM_{\mu_2}(i)$, $\cdots$, $DOM_{\mu_k}(i)$, respectively, and furthermore become a group of *Parallel Links* if their HTML tags are 'hyperlinks' (*i.e.*, <a>). Finally, we scan pages in P one by one, and extract all groups of parallel links with the size no less than 4. The similarity $Sim_P(i,j)$ of pages p_i and p_j gained from in-page link-structures equals to how many times p_i and p_j appear in a group of parallel links.

3.3 Consolidating with Content-Based Similarities

The final similarity $Sim(i,j)$ for pages p_i and p_j is:

$$SIM(i,j) = SIM_{CB}(i,j) + w_2 \cdot SIM_P(i,j) + w_1 \cdot SIM_{content}(i,j) \qquad (2)$$

Here $SIM_{content}(i,j)$ can be obtained by any kind of content-based similarity functions [6] [20]. w_1 and w_2 are parameters that tunes linear weights among the three parts. Different values of w_1 and w_2 express different emphasis to structure-based and content-based similarities. There could be more than one good answers for a page clustering task. It is not a competition between two runners (*i.e.*, structure-based and content-based similarities) to see which one has the better performance, instead we are installing two engines for more effective similarity functions as well as clustering results.

4 Hierarchical Clustering

Although there have been many clustering algorithms developed for web applications, we choose to further develop the density-based approach with the following reasoning.

1. Web clusters may have arbitrary shapes and the data points inside a cluster may be arbitrarily distributed. Density-based clustering is good at generating clusters of arbitrary shapes.
2. Web datasets are usually huge. Density-based clustering can be linear to the number of edges. Moreover, since the average number of hyper-links inside one page is regarded as a constant, the number of edges is approximately linear to the number of vertices.
3. Web clusters may vary a lot in size, and web datasets contain noises. Spectral partitioning algorithms have constraints on cluster sizes, and agglomerative hierarchical clustering methods are sensitive to outliers.

[2] Two sub-trees are siblings if they have the same parent.

SCAN [17] is a one-level density-based network clustering algorithm, of which one clear advantage is its linear time complexity that out-performs other methods. However, SCAN requires two parameters, and the optimal parameters that lead to the best clustering performance are given by human via visualization. In this section, we extend SCAN to a hierarchical clustering algorithm, called HSClus.

Algorithm Framework. It is natural to derive HSClus from SCAN by iteratively applying SCAN to each cluster obtained by SCAN in the previous step. However, since different sets of pages may have different optimal parameters, it is infeasible to select two fixed parameters as the input for each call of SCAN. To solve this problem, HSClus (i) tests SCAN with different pairs of parameters, (ii) uses a scoring function to evaluate the clustering results under different parameters, and (iii) finally clusters pages by the optimal parameters. For each resulting cluster, HSClus repeats the same procedure until termination conditions are met. Because of the space limitation, details are omitted.

Complexity Analysis. Usually, the number of levels L in a clustering hierarchy is small (*e.g.*, no more than 10), and we select a constant number (say K) of parameters to test. Since SCAN is linear to the number of edges m, the time complexity of HSClus is also linear, which is $O(LKm)$.

5 Experiments

In this section, we evaluate the efficiency and the effectiveness of HSClus on both synthetic and real datasets. All algorithms are implemented in Java Eclipse and Microsoft Visual Studio 2008, conducted in a PC with 1.5GHz CPU and 3GB main memory. We compare HSClus with two algorithms: (i) *k-medoids* [13] and (ii) *FastModularity* [3].

5.1 Effectiveness

A real dataset $UIUC\ CS$ is the complete set of pages in the domain of $cs.uiuc.edu$ crawled down by Oct. 3, 2008. It has $12,452$ web-pages and $122,866$ hyper-links. The average degree of each page is 19.7, and $33,845$ groups of parallel links are discovered.

To evaluate the usefulness of parallel links, *Figue.* 1 and 2 show the clustering results with and without similarities gained from in-page link structure. As observed, the two figures are generally the same at high levels; in low levels, *Figue.* 2 may mix pages that are in different kinds but have close semantic meanings, *e.g.*, $Research/Faculty$ and $Research/Area$ alternate, and $Undergraduate/Transfer$ is in the middle of $Undergraduate/Course$.

Figue. 3 and 4 are the results generated by FastModularity and k-medoids (some parts are omitted), respectively. We can observe that, the clustering quality is much lower than HSClus.

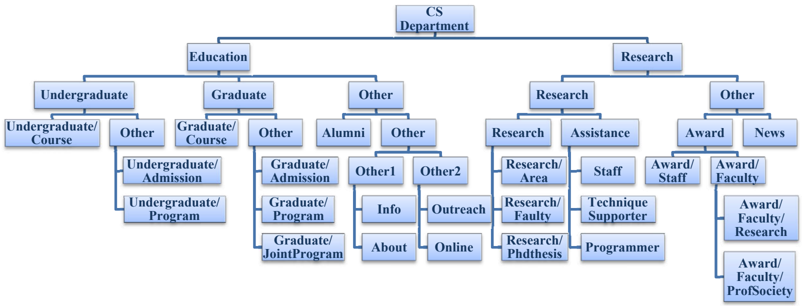


Fig. 1. Result of HSClus with the similarities gained from parallel links

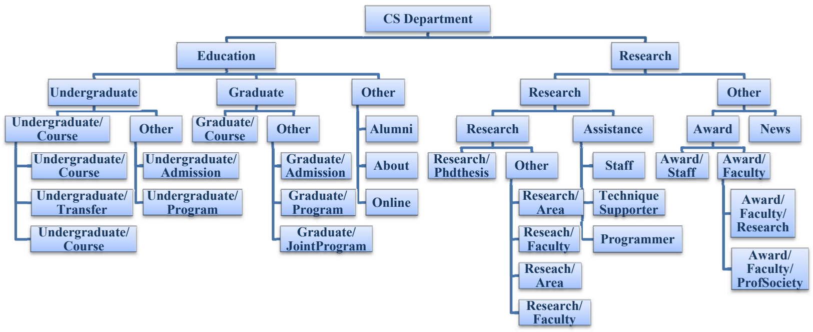


Fig. 2. Result of HSClus without the similarities gained from parallel links

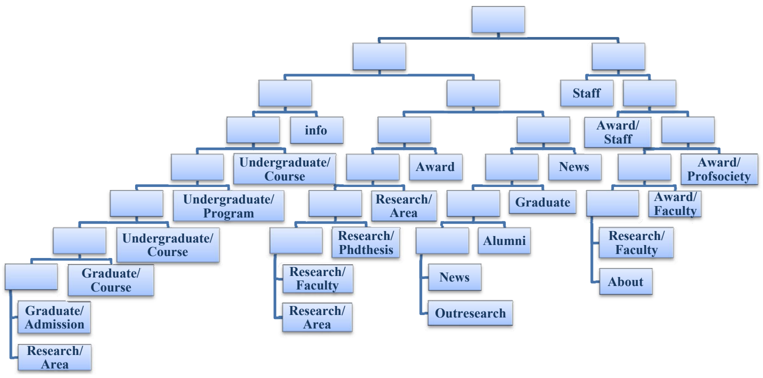


Fig. 3. Result of FastModularity on UIUC CS

5.2 Efficiency

To verify that HSClus is as fast as linear against networks of different sizes, we generate 8 synthetic graphs, whose numbers of edges range from 2, 414 to 61, 713, 102, to test corresponding running times. We can see in *Figue.* 5 that

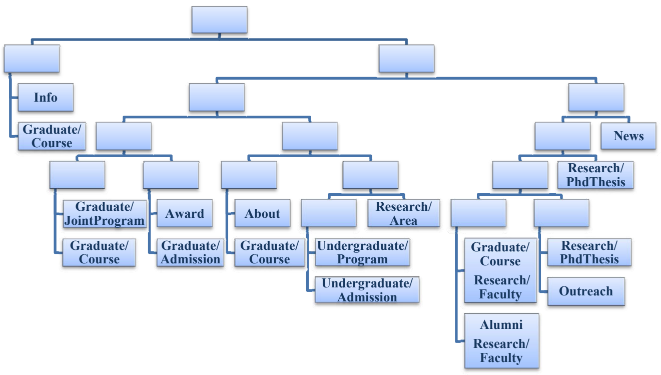


Fig. 4. Result of k-medoids on UIUC CS

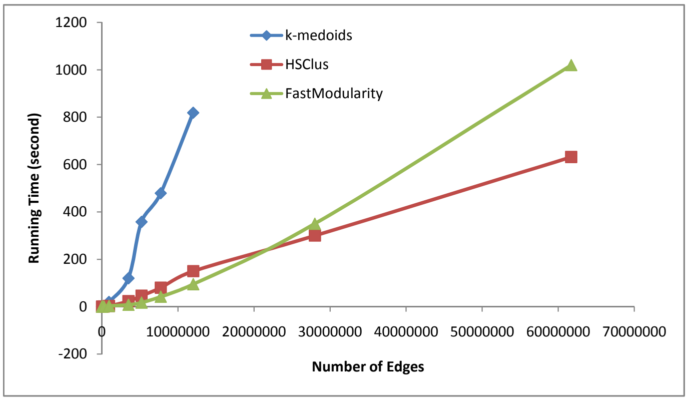


Fig. 5. Efficiency comparison

the running time (in second) of HSClus is linear against the input size (number of edges); FastModularity increases more quickly than linear; and k-medoids rises dramatically.

6 Conclusion

This paper develops a novel method for hierarchical clustering of webpages in an organization in order to discover the inherent semantic structure of the website. Both cross-page link structure and in-page link organizations are explored to produce a new similarity function, and a new density-based clustering algorithm is developed to group densely linked webpages into semantic clusters and identifies their hierarchical relationships. Our experiments show that this method is efficient and effective at uncovering webpage structures at some organizations websites and sheds light on mining and exploring web semantic structures.

References

1. Bekkerman, R., Zilberstein, S., Allan, J.: Web page clustering using heuristic search in the web graph. In: IJCAI (2007)
2. Chehreghani, M.H., Abolhassani, H., Chehreghani, M.H.: Improving density-based methods for hierarchical clustering of web pages. Data Knowledge Engineer (2007)
3. Clauset, A., Newman, M.E.J., Moore, C.: Finding community in very large networks. Physical Review (2004)
4. Crescenzi, V., Merialdo, P., Missier, P.: Clustering web pages based on their structure. Data Knowledge Engineer (2005)
5. Ding, C.: Spectral clustering tutorial. In: ICML (2004)
6. Glassman, C., Manasse, M., Zweig, G.: Syntactic clustering of the web. In: WWW (2005)
7. Hou, J., Zhang, Y.: Utilizing hyperlink transitivity to improve web page clusterings. In: ADC (2003)
8. Kessler, M.: Bibliographic coupling between scientific papers. American Documentation (1963)
9. Kleinberg, J.: Authoritative sources in a hyperlinked environment. Journal of ACM-SIAM (1998)
10. Kudelka, M., Snasel, V., Lehecka, O., El-Qawasmeh, E., Pokorny, J.: Web pages reordering and clustering based on web patterns. In: Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., Bieliková, M. (eds.) SOFSEM 2008. LNCS, vol. 4910, pp. 731–742. Springer, Heidelberg (2008)
11. Manning, C.D., Schützen, H.: Foundations of statistical natural language processing. MIT Press, Cambridge (1999)
12. Milligan, G., Cooper, M.: A study of the comparability of external criteria for hierarchical cluster analysis. Multivariate Behavioral Research (1986)
13. Ng, R., Han, J.: Efficient and effective clustering methods for spatial data mining. In: VLDB (1994)
14. Qi, X., Davison, B.: Knowing a web page by the company it keeps. In: CIKM (2006)
15. Small, H.: Co-citation in the scientific literature: A new measure of the relationship between two documents. American Social Info. Science (1973)
16. Wu, T., Chen, Y., Han, J.: Association mining in large databases: A re-examination of its measures. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 621–628. Springer, Heidelberg (2007)
17. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.J.: Scan: a structural clustering algorithm for networks. In: KDD (2007)
18. Yi, O.: Ehm-based web pages fuzzy clustering algorithm. In: MUE (2007)
19. Yitong Wang, M.K.: Evaluating contents-link coupled web page clustering for web search results. In: CIKM (2002)
20. Zamir, O., Etzioni, O.: Web document clustering: a feasible demonstration. In: SIGIR (1998)

Mining Numbers in Text Using Suffix Arrays and Clustering Based on Dirichlet Process Mixture Models

Minoru Yoshida[1], Issei Sato[1], Hiroshi Nakagawa[1], and Akira Terada[2]

[1] University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo 113-0033
{mino,sato,nakagawa}@r.dl.itc.u-tokyo.ac.jp
[2] Japan Airlines, 3-2, Haneda Airport 3-Chome, Ota-ku, Tokyo 144-0041 Japan
akira.terada@jal.com

Abstract. We propose a system that enables us to search with ranges of numbers. Both queries and resulting strings can be both strings and numbers (e.g., "200–800 dollars"). The system is based on suffix-arrays augmented with treatment of number information to provide search for numbers by words, and vice versa. Further, the system performs clustering based on a Dirichlet Process Mixture of Gaussians to treat extracted collection of numbers appropriately.

Keywords: Number Mining, Suffix Arrays, Dirichlet Process Mixture, Clustering.

1 Introduction

Texts often contain a lot of numbers. However, they are stored simply as strings of digits in texts, and it is not obvious how to treat them as not strings but numeric values. For example, systems that treat numbers simply as strings of digits have to treat all numbers "1", "2", "213", and "215" as different, or all of them in the same way (e.g., by replacing them with "0"). In this paper, we propose treating numbers more flexibly, such as *similar* numbers like "1" and "2" should be treated as the same, "213" and "215" should also be treated the same, but "1" and "213" should be treated as different. *Range of numbers* is a representation of number collections that is appropriate for this purpose. In the above case, the collection of "1" and "2" can be expressed by the range "1..2" and the collection of "213" and "215" can be expressed by "213..215". Not only it can represent a lot of numbers compactly, but also it covers the numbers similar to the given collections not found in the given collection.

We propose a system that provides the following two basic indispensable functions for treating a range of numbers as normal strings:

– the function to derive appropriate number ranges from a collection of numbers,
– the function to search texts by using number range queries.

The former is to *find* the range of numbers inherent in a collection of numbers and the latter is to *use* the extracted number ranges for further processing.

For the former problem of finding number ranges, the system dynamically clusters the numbers in the search results based on the Dirichlet process mixture (DPM) [1] clustering algorithm, which can automatically estimate the appropriate number of clusters. Our DPM model for number clustering is a mixture of Gaussians [2], which is a very popular example of DPM models. Inference for cluster assignment for DPM models has been extensively studied for many previous papers, including MCMC [3], variational Bayes [4], and A* or beam search [5]. However, our task is somewhat different from the ones discussed in these papers, because our task is to derive appropriate *number ranges*, which require constraints to be put on the derived cluster assignments that each cluster must consist of contiguous regions, and it is unobvious how to incorporate them into existing inference algorithms. To the best of our knowledge, no previous studies have discussed how to derive such number ranges on DPM models.

For the latter problem of the number range search, we propose using *suffix arrays* for the basic index structures. We call the resulting index structure *number suffix arrays*. The following operations are possible on number suffix arrays.

TF calculation: obtaining the counts for the queries that contain the range of numbers.

Adjacent string calculation: obtaining the strings (or tries) next to the range of numbers.

Search engine providers are one of many groups that have extensively studied indexing for searching by range of numeric values. Fontoura et al. [6] proposed an indexing method with inverted-indexes to efficiently restrict a search to the range of values of some of the numeric fields related to the given documents. In particular, Google search ("search by numbers") in English provides a search option ".." to indicate the number ranges. The inverted-index based methods for number range retrieval are for returning the *positions* of the numbers in the given range. On the other hand, our number suffix arrays not only return the positions, but also return the suffix array for the strings adjacent to the query, which can be used as a trie of the strings adjacent to the query and can be used for many text mining applications. These applications include extracting frequent adjacent string patterns for further text mining operations like synonym extraction (as shown in the later sections). In other words, number suffix arrays can be regarded as the extended version of the normal indexes for number ranges that are more appropriate for text mining tasks.

2 Number Suffix Arrays

The main component of our number mining system is *number suffix arrays*, which are based on suffix arrays [7] and can enable searches by numbers. Suffix arrays are data structures that represent all the suffixes of a given string. They are sorted arrays of (positions of) all suffixes of a string. By use of the suffix array constructed from the corpus S, all the positions of any substring s of S can be obtained quickly (in $O(|s|\log|S|)$ time, where $|x|$ is the length of x) for any s by using binary search on the suffix array. Suffix arrays require $2|S|$ bytes[1] of additional space to store indices and even more space for construction. We assume that both the corpus and the suffix array are stored in memory. We denote by $S[i..]$ the suffix of S starting from index i.

Our algorithm for searching for a range of numbers is defined as follows. Assume that the input query is a string $s_1.[lb_1..ub_1].s_2.[lb_2..ub_2]...s_n$ where "." means concatenation of adjacent strings, and lb_k and ub_k are integers. Strings surrounded by `[` and `]` in a query represent the range of numbers from the number on the left of `..` to the number on the right of `..`. For the query q =`KDD-[2000..2005]`, s_1=`KDD-`, $lb_1 = 2000$, $ub_1 = 2005$, and s_2="" (null string), where $n = 2$. Setting the current index array $ca = sa$ (sa is a suffix array of the whole input document), our algorithm iterates the following steps for $k = 1, 2, ..., n$.

1. Search for string s_k on the array ca, and obtain the resulting range of indices $[x \ldots y]$. Create a new array sa_2 of strings adjacent to s_k by letting $sa_2[i] = sa[x + i] + |s_k|$[2]. Let $ca = sa_2$.
2. Search for all digits that follow s_k. This is done by searching for the index of the character `0` on ca, and obtain the resulting index i_1, and in the same way, searching for the index of the character `9` on ca, and obtain the resulting index i_2. For each $i_1 \leq j \leq i_2$, parse the consecutive digits in the prefix of $S[sa_2[j]..]$ (suffixes of S starting from the position $sa_2[j]$), and obtain the resulting value d. If $lb_k \leq d \leq ub_k$, add the index i_3 (i_3: index of the end of the digits) to a new array sa_3. [3]
3. Sort the array sa_3 according to the alphabetic order of $S[sa_3[j]..]$. Let $ca = sa_3$.

In general, the range $[i_1 \ldots i_2]$ in step-2 in the above algorithm will not be so large because it is only covers the suffixes that are at least preceded by s_1 and start with digits. However, if s_1 is null (*i.e.,* the query starts by the range of numbers such as `[100..200] years old`), the range $[i_1 \ldots i_2]$ will be considerably large (it will be the number of all numbers in the text), which means scanning the range will be prohibitively time-consuming. Our basic idea to solve this problem is to make an additional array, which we call a *number array*, that retains *numeric ordering.* The number array na for corpus S is the array of indices that all point to the start point of all consecutive digits in S. It is sorted by the numeric order of the numbers represented by the digits that start from each position pointed to by the indices, and we can find (ranges of) numbers by performing binary search on this array with numeric-order comparison.

[1] This is if each index is represented by four bytes and each character takes two bytes.
[2] Here, $|s|$ is the length of string s.
[3] We do not use an approach to modify the corpus by replacing numbers with one character representing the value because it reduces the system's ability in some cases, e.g., it will limit variety of possible values to $2^{sizeof(char)}$, disable digit-pattern-matching queries such as "Boeing 7*7", etc.

3 Number Clustering by Dirichlet Process Mixture Models

Search results for number suffix arrays also may contain numbers. Number suffix arrays can describe collections of different numbers by number ranges, i.e., by the smallest and largest values, such as "[20..50]" for the collection of 20, 23, 30, 35, 42, and 50. The problem here is that these values do not always appropriately represent the collection. For instance, expressing the collection $< 1, 2, 3, 4, 1000, 1001, 1002 >$ as [1..1002] loses the information that values in the collection are concentrated in two ranges (*i.e.*, [1..4] and [1000..1002]). This problem can be avoided by dividing the collection into clusters.

Clustering algorithms that need to set the number of clusters (*e.g.*, K-means) are not appropriate for our situation because the appropriate number of clusters is different for each collection of numbers. Of the clustering algorithms that do not need data on the number of clusters, we selected the DPM [1] clustering algorithm because it provides the principles to probabilistically compare clustering results even if the number of clusters differs among distinct clustering results.

Given the collection of numbers $x_1, \cdots, x_n$[4], assume there exists the hidden parameter θ_i for each number x_i. The Dirichlet process [8] is a distribution over distributions and generates a discrete (with probability one) distribution over a given set (which is all the real numbers in our case). Let G be a distribution drawn from the Dirichlet process. Each value θ_i is drawn from G, where each observation x_i is generated from a distribution with parameter θ_i: $G \sim DP(\alpha, G_0)$, $\theta_i \sim G$, and $x_i \sim f(\theta_i)$. The parameters of the Dirichlet process are base distribution G_0 and concentration parameter α.

The Dirichlet process can give the probability of *clusters of* θ_i when G is integrated out. Here, θ_i and θ_j (, and thus x_i and x_j) are in the same cluster if $\theta_i = \theta_j$. Let C_j be a cluster of indices $c_{j1}, c_{j2}, \cdots, c_{j|C_j|}$ so that $\theta_{c_{j1}} = \theta_{c_{j2}} = \cdots = \theta_{c_{j|C_j|}}$. We denote the collection of all C_j as C. Then, the probability of the collection of θ_i is given as $p(\boldsymbol{\theta}) = \frac{\alpha^{|C|}}{\alpha^{(n)}} \prod_{j=1}^{|C|} G_0(\theta'_j)(|C_j| - 1)!$ where $|C|$ is the number of clusters, $|C_j|$ is the number of observations in the jth cluster and $\alpha^{(n)} = \alpha(\alpha + 1)(\alpha + 2) \cdots (\alpha + n - 1)$. θ'_j is the value for the jth cluster (*i.e.*, $\theta'_j = \theta_{j1} = \theta_{j2} = \cdots = \theta_{j|C_j|}$).

We use a DPM of Gaussians (or, equivalently, the infinite Gaussian mixture [2]) model. In our model, both G_0 and f are assumed to be Gaussians, with (mean,deviation) being (μ_1, σ_1) for the former and (θ_i, σ_2) for the latter: $G_0 = \mathcal{N}(\mu_1, \sigma_1)$, and $f_i = \mathcal{N}(\theta_i, \sigma_2)$.[5]

[4] We use the logarithm of each number in search results as x_i, which is based on the observation that relative sizes are appropriate as similarities between numbers rather than as absolute difference values.

[5] σ_1, σ_2 and μ_1 are fixed to reduce computation time. We set μ_1 to 0 and σ_1 to 100.0 to resemble the uniform distribution for the prior probability of θ_i to minimize bias in the value of θ_i. Other parameters are currently set to $\alpha = 1.0$ and $\sigma_2 = 1.0$.

The joint distribution of $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ and $\boldsymbol{\theta}$ is thus

$$p(\boldsymbol{x}, \boldsymbol{\theta}) = \frac{\alpha^{|C|}}{\alpha^{(n)}} \prod_{j=1}^{|C|} (|C_j| - 1)! \frac{1}{\sqrt{2\pi}\sigma_1} exp\left\{ \frac{-(\theta'_j - \mu_1)^2}{2\sigma_1^2} \right\} \prod_{i=1}^{|C_j|} \frac{1}{\sqrt{2\pi}\sigma_2} exp\left\{ \frac{-(x_{c_{ji}} - \theta'_j)^2}{2\sigma_2^2} \right\}$$

We integrate out θ' because we need only cluster assignments, not parameter values themselves. This results in the the objective function to maximize (which indicates the goodness of clustering), which is denoted by $f(C)$.

$$f(C) = \frac{\alpha^{|C|}}{\alpha^{(n)}} \frac{1}{\sqrt{2\pi}^n \sigma_2^n} \prod_{j=1}^{|C|} g(C_j) \tag{1}$$

where

$$g(C_j) = (|C_j| - 1)! \frac{1}{\sqrt{1 + |C_j|(\frac{\sigma_1}{\sigma_2})^2}} exp\left\{ -\frac{1}{2\sigma_2^2} \left\{ (\sum_{i=1}^{|C_j|} x_{c_{ji}}^2) - \frac{\sigma_1^2}{\sigma_2^2 + |C_j|\sigma_1^2} (\sum_{i=1}^{|C_j|} x_{c_{ji}})^2 \right\} \right\}$$

The algorithm searches for the best cluster assignment that maximizes the objective function (1). Note that the objective function (1) is defined as the product of g for each cluster, which means that the function is "context-free" in the sense that we can independently calculate score $g(C_j)$ and then multiply it to calculate f because the value of $g(C_i)$ is not affected by changes in other clusters C_i s.t. $i \neq j$. Note that our purpose in clustering is to appropriately divide a given number collection into *continuous regions*. Therefore, we do not need to consider the case where a cluster is not a region (*i.e.,* elements in the cluster are separated by elements in another cluster, such as a case where $C_1 = \{1, 5\}$ and $C_2 = \{2, 6\}$.)

In this situation, the best cluster assignment can be found by a naive dynamic programming approach. We call this approach the *baseline algorithm* or *CKY algorithm* because it is a bottom-up style algorithm performed in the same way as the Cocke-Younger-Kasami (CKY)-parsing algorithm for context free grammar, which is popular in the natural language processing community.

In our approach, we accelerate the search further by using a greedy search strategy. Starting from no partition (*i.e.,* all elements are in the same region (cluster)), the algorithm divides each region into two sub-regions to best increase the objective function (1) and then recursively divides the sub-regions. If it is not possible to divide a region into two sub-regions without decreasing the objective function value, division stops.

More precisely, number clustering is done by calling the following function *partition*(A), where A is the collection of all numbers input to the algorithm. After that, we obtain C as the clustering result.

Partition(N)**:** Find the best partition $left'(N)$ and $right'(N)$ that maximizes $g(left(N))g(right(N))$. If $\alpha \cdot g(left'(N))g(right'(N)) \leq g(N)$, then add N to C (*i.e.*, partitioning of N stops and N is added to the resulting cluster set). Otherwise, call $partition(left'(N))$ and $partition(right'(N))$ recursively.

Here, α is multiplied with $g(left'(N))g(right'(N))$ because partitioning increases the number of clusters $|C|$ that appear as $\alpha^{|C|}$ in objective function (1).

4 Experiments: Synonym Extraction

Our flexible number handling is useful in many text-mining tasks, especially when we want to use the numbers as some kind of *contexts*. A typical example is measuring the semantic similarities of words. When measuring word similarities, we typically calculate the *distributions* of words related to word w (e.g., distribution of words around w, distribution of words that have dependency relations with w, etc.) as the contexts of w, and measure the similarities of the meanings of the words w_1 and w_2 by calculating the similarities of their contexts.

A direct application of measuring similarities of words is *synonym extraction*. Especially, we developed an algorithm to dynamically extract synonyms of given queries using suffix arrays [9]. To find words similar to a given query q, the algorithm extracts context strings (*i.e.*, strings that precede or follow q) by using suffix arrays[6], which in turn are used to find strings surrounded by these extracted contexts.

We enhanced the algorithm by adding the ability to appropriately treat numbers in context strings in number suffix arrays. For example, we can use the context strings "[10..20] persons" to cover all numbers between 10 and 20 preceding the word "persons", while in naive suffix arrays, only raw strings such as "11 persons" and "17 persons" can be used as contexts. Our number suffix arrays can thus improve coverage of contexts and extracted collections of synonyms.

We evaluate the performance of synonym extraction with number suffix arrays to investigate whether number suffix arrays enhance text mining. We used aviation-safety-information texts from Japan Airlines that had been de-identified for data security and anonymity. The reports were in Japanese, except for some technical terms in English. The size of the concatenated documents was 6.1 Mbytes. Our text-mining system was run on a machine with an Intel Core Solo U1300 (1.06 GHz) processor and 2 GByte memory. All algorithms were implemented in Java. The size of the number array for each (normal or reversed) suffix array was 60,766.

To evaluate the performance of the system, we used a thesaurus for this corpus that was manually developed and independent of this research. The thesaurus consists of $(t, S(t))$ pairs, where t is a term and $S(t)$ is a set of synonyms of t. We provided t as a query to the system, which in turn returned a list of synonym candidates $\langle c_1, c_2, ..., c_n \rangle$ ranked on the basis of their similarities to the query. $S(t)$ was used as a correct answer to evaluate the synonym list produced by the system. The number of queries was 404 and the average number of synonyms was 1.92.

We compared the average precision [10] of our algorithm with the baseline (using naive suffix arrays) and the no-clustering version (using number suffix arrays

[6] We use two suffix arrays: one is a normal suffix array and the other is a *reversed suffix array*, which is a suffix array constructed from the reversed original text.

Table 1. Results of Synonym Extraction. σ_1 was set to 100.0. AvPrec is the average precision (per cent) and Time is the average execution time (per second) for each query.

Algorithm	AvPrec	Time
Baseline	40.66	2.294
No Clustering	41.30	10.272
Our Algorithm	41.68	7.837

Table 2. (Left:) Execution Time (Per Second) for Number Queries. NumStart is for Queries that Start with Number Ranges, and NotNumStart is for Queries that Start with Non-digit Characters. (Right:) Execution time (per second) and total log likelihood of number clustering.

Algorithm	NumStart	NotNumStart
Baseline	169.501	0.711
w/ Number Arrays	12.87	0.632

Algorithm	Time	Log Likelihood
CKY	102.638	-168021.7
Greedy	0.170	-168142.2

without number clustering). The results are shown in Table 1. We observed that the performance was improved by using number suffix arrays by about 0.6 percent, which was improved further by about an additional 0.4 percent by performing number clustering. However, the average execution time for each query became 3.5–4.5 times larger than that of the baseline. For practical use, we will have to reduce the execution time by reducing the number of range-starting queries (*i.e.*, the queries that start with a range of numbers).

4.1 Results: Speed and Accuracy of the Algorithm

We stored all the queries to the number suffix arrays and all the collections of numbers for number clustering that appeared in the above experiments. We randomly selected 200 queries that included the range of numbers for each suffix array (normal and reversed), resulting in 400 queries in total. Of each 200 queries, 100 started with a range of numbers (indicated as "NumStart"), and the remaining 100 started with non-digit characters (indicated as "NotNumStart"). We also randomly selected 1000 collections of numbers, and used them to measure the accuracy and execution time of our number clustering algorithms.[7]

The result of the query-time experiment is shown in Table 2 (left). We observed that the search time for queries starting with a range of numbers was drastically reduced by using the number arrays. Considering the large ratio of the search time of NumStart and NotNumStart, using the number arrays is an efficient way to conduct a number search.

The results of a comparison of two clustering algorithms are shown in Table 2 (right). The greedy algorithm was much faster than the baseline CKY algorithm. The important point here is that the difference of total log-likelihood values

[7] Only collections whose sizes were from 50 to 1000 were selected. The average size of collections was 98.9.

between the greedy (approximate) algorithm and the baseline was quite small, which suggests that using the greedy algorithm for number clustering achieves much faster processing with almost no sacrifice of quality of the clustering results.

5 Conclusion and Future Work

We described number suffix arrays, which enable us to search for numbers in text. The system is based on suffix arrays and DPM clustering. We also showed applications of number suffix arrays to text mining, including synonym extraction, where the performance could be improved by using number suffix arrays. Future work includes developing more sophisticated preprocessing for numbers such as normalization of number expressions.

Acknowledgement

This research was partially supported by Fujitsu Laboratories Ltd.

References

1. Antoniak, C.E.: Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. The Annals of Statistics 2(6), 1152–1174 (1974)
2. Rasmussen, C.E.: The infinite Gaussian mixture model. In: Advances in Neural Information Processing Systems, 13th Conference, NIPS 1999, pp. 554–560 (2000)
3. Jain, S., Neal, R.M.: Splitting and merging components of a nonconjugate dirichlet process mixture model. Technical Report 0507, Dept. of Statistics, University of Toronto (2005)
4. Blei, D.M., Jordan, M.I.: Variational inference for dirichlet process mixtures. Bayesian Analysis 1(1), 121–144 (2006)
5. Daumé, H.: Fast search for Dirichlet process mixture models. In: Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 83–90 (2007)
6. Fontoura, M., Lempel, R., Qi, R., Zien, J.Y.: Inverted index support for numeric search. Internet Mathematics 3(2), 153–186 (2006)
7. Manber, U., Myers, G.: Suffix arrays: A new method for on-line string searches. In: Proceedings of the first ACM-SIAM Symposium on Discrete Algorithms, pp. 319–327 (1990)
8. Ferguson, T.S.: A Bayesian analysis of some nonparametric problems. The Annals of Statistics 1(2), 209–230 (1973)
9. Yoshida, M., Nakagawa, H., Terada, A.: Gram-free synonym extraction via suffix arrays. In: Li, H., Liu, T., Ma, W.-Y., Sakai, T., Wong, K.-F., Zhou, G. (eds.) AIRS 2008. LNCS, vol. 4993, pp. 282–291. Springer, Heidelberg (2008)
10. Chakrabarti, S.: Mining the Web: Discovering Knowledge from Hypertext Data. Morgan-Kaufmann Publishers, San Francisco (2002)

Opinion-Based Imprecise Query Answering

Muhammad Abulaish[1,*], Tanvir Ahmad[2], Jahiruddin[1], and Mohammad Najmud Doja[2]

[1] Department of Computer Science, Jamia Millia Islamia, New Delhi, India
[2] Department of Computer Engineering, Jamia Millia Islamia, New Delhi, India
{abulaish.cs,tanvir.ce}@jmi.ac.in,
jahir.jmi@gmail.com, ndoja@yahoo.com

Abstract. There is an exponential growth in user-generated contents in the form of customer reviews on the Web. But, most of the contents are stored in either unstructured or semi-structured format due to which distillation of knowledge from this huge repository is a challenging task. In addition, on analysis we found that most of the users use fuzzy terms instead of crisp terms to express opinions on product features. Considering these facts, in this paper, we present an opinion-based query answering framework which mines product features and opinionated words to handle user queries over review documents. The proposed framework uses BK-FIRM (Bandler-Kohout Fuzzy Information Retrieval Model) that facilitates the formulation of imprecise queries using linguistic qualifiers, retrieves relevant opinion documents, and presents them in the order of their degree of relevance. The efficacy of the system is established through experiments over customer reviews on different models of digital camera, and mp3 player.

Keywords: Opinion Mining, Sentiment Analysis, Opinion-Based Query Answering, Imprecise Query Processing, Natural Language Processing.

1 Introduction

Due to easy accessibility of Web, numerous forums, discussion groups, and blogs exist and individual users are participating more actively and are generating vast amount of new data – termed as *user-generated contents*. These new web contents include customer reviews and blogs that express opinions on products and services – which are collectively referred to as customer feedback data on the Web. As customer feedback on the Web influences other customer's decisions, these feedbacks have become an important source of information for businesses to take into account when developing marketing and product development plans. Now much of the information is publicly available on the Web. As a result, the number of reviews that a product receives grows rapidly. Some popular products can get hundreds of reviews or more at some large merchant sites. Many reviews are also long, which makes it hard for potential customers to read them to make an informed decision on whether to purchase the product. A large number of reviews for a single product may also make

[*] Corresponding author.

it harder for individuals to evaluate the true underlying quality of a product. In these cases, customers may naturally gravitate to reading a *few reviews* in order to form a decision regarding the product and he/she only gets a biased view of the product. Similarly, manufacturers want to read the reviews to identify what elements of a product affect sales most. And, the large number of reviews makes it hard for product manufacturers or business to keep track of customer's opinions and sentiments on their products and services. Recent work has shown that the distribution of an overwhelming majority of reviews posted in online markets is bimodal [7]. Reviews are either allotted an extremely high rating or an extremely low rating. In such situations, the average numerical star rating assigned to a product may not convey a lot of information to a prospective buyer. Instead, the reader has to read the actual reviews to examine which of the positive and which of the negative aspect of the product are of interest. Several sentiment analysis approaches have proposed to tackle this challenge up to some extent. However, most of the classical sentiment analysis mapping the customer reviews into binary classes – *positive or negative*, fails to identify the product features liked or disliked by the customers.

In this paper, we present an opinion-based query answering framework that mines product features and opinionated words from opinion texts. The proposed framework uses BK-FIRM (Bandler-Kohout Fuzzy Information Retrieval Model) to handle opinion-oriented imprecise user queries over review documents. Linguistic and semantic analyses are applied to identify key information components that are centered on product features. Since, on analysis we found that most of the users use fuzzy terms instead of crisp terms to express opinions on product features, an information component is defined as a triplet $<\mathcal{F}, \mathcal{M}, \mathcal{O}>$ where, $\mathcal{F}$ represents a product feature, $\mathcal{O}$ represents opinion words associated with $\mathcal{F}$ and $\mathcal{M}$ is an optional component representing adverbs that act as modifier and used to intensify the opinion $\mathcal{O}$. $\mathcal{M}$ is also used to capture the negative opinions explicitly expressed in the review. The novelty of the system lies in mining associated modifiers with opinions. For example, consider following snippets of opinion sentences: *(i) the picture quality is very good; (ii) the picture quality is almost good.* In both of the sentences the opinion word is *good* but the associated modifiers are different to express different levels of customer satisfaction on picture quality. For each extracted feature, the list of opinions and associated modifiers are compiled and stored in a structured repository to answer user query over it.

The remaining paper is structured as follows: Section 2 presents a brief review on opinion mining. It also presents the overview of the BK-FIRM model and its working principles. In section 3, we present the opinion-based query answering framework. The experimental setup and evaluation results are presented in section 4. Finally, section 5 concludes the paper with possible enhancements to the proposed system.

2 Related Work

In this section, we present a summarized view of the existing works on opinion mining and sentiment analysis which is followed by a brief introduction of the BK-FIRM model and its working principles.

2.1 Opinion Mining and Sentiment Analysis

Research on opinion mining started with identifying opinion bearing words, e.g., *great*, *amazing*, *wonderful*, *bad*, *poor* etc. In literature, a reasonable number of attempts have been made to mine such words and identifying their semantic orientations [3,5]. The history of the phrase *"sentiment analysis"* parallels that of *opinion mining* in certain respects. A sizeable number of papers mentioning sentiment analysis focus on the specific application of classifying customer reviews as to their polarity – positive or negative [10,11]. Although, classical sentiment classification attempts to assign the review documents either *positive* or *negative* class, it fails to find what the reviewer or opinion holder likes or dislikes. A positive document on an object does not mean that the opinion holder has positive opinions on all aspects or features of the object. Likewise, a negative document does not mean that the opinion holder dislikes everything about the object. In an evaluative document (e.g., *a product review*), the opinion holder typically writes both positive and negative aspects of the object, although the general sentiment on the object may be positive or negative. To obtain detailed aspects, feature-based opinion mining is proposed in literature [3,4,6]. In [4], a supervised pattern mining method is proposed. In [3, 6], an unsupervised method is used. A lexicon-based approach has been shown to perform quite well in [2, 3]. The lexicon-based approach basically uses opinion words and phrases in a sentence to determine the orientation of an opinion on a feature.

Although, some opinion mining methods extract features and opinions from document corpora, most of them do not explicitly exploit the semantic relationships between them. The proposed method differs from all these approaches predominantly in its use of pure linguistic techniques to identify only those features for which customers have commented using opinionated words. Extraction of associated modifiers used in review documents to represent the degree of expressiveness of opinions is unique in our work. Moreover, to the best of our knowledge, none of the above-cited works attempted to use the mined features and opinions for query answering.

2.2 BK-FIRM

Different from traditional information retrieval theories, BK-FIRM uses the concept of fuzzy relation to retrieve documents based on semantics and it has basic functions such as automated building of a thesaurus and ranking the retrieved documents. BK-FIRM has two operations, (i) R-request operation which expands semantics of a term, and (ii) FS-request operation which analyzes user query and retrieves documents relevant to the given query [1]. The procedure of BK-FIRM is as follow. Assume that there are a document set $D=\{d_1, d_2, \ldots, d_k\}$, a term set $T=\{t_1, t_2, \ldots, t_n\}$ and a fuzzy relation $\mathfrak{R}_F$ (equation 1) between the document set and the term set. When query Q as FS-request is given from a user, the fuzzy relation $\mathfrak{R}_F$ applied to the query Q gets a fuzzy set D_F (equation 2), which means the suitability of the query Q to document set. Then, an α-cut is applied to the fuzzy set D_F to get a resultant document set D_R (equation 3).

$$\Re_F = D \times T = \begin{array}{cccc|c} v_{11} & v_{12} & \cdots & v_{1n} & d_1 \\ v_{21} & v_{22} & \cdots & v_{2n} & d_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ v_{k1} & v_{k2} & \cdots & v_{kn} & d_k \\ \mathrm{t}_1 & t_2 & \cdots & t_n & \end{array} \tag{1}$$

$$D_F = Q(\Re_F) = Q(\begin{array}{cccc|c} v_{11} & v_{12} & \cdots & v_{1n} & d_1 \\ v_{21} & v_{22} & \cdots & v_{2n} & d_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ v_{k1} & v_{k2} & \cdots & v_{kn} & d_k \\ \mathrm{t}_1 & t_2 & \cdots & t_n & \end{array}) = \{d_1/v_1, d_2/v_2, \cdots, d_k/v_k\} \tag{2}$$

$$D_R = \{d_1^{'}, d_2^{'}, \cdots, d_k^{'}\} \tag{3}$$

3 Proposed Framework

Fig. 1 presents the architectural details of the proposed opinion-based query answering framework which consists of two major modules – *Feature and Opinion Learner, and Opinion-Based Query Processor.* The working principles of these components are explained in the following sub-sections.

3.1 Feature and Opinion Learner

In this section, we present the working details of the feature and opinion learner module which completes its task in the following three steps (i) Document processing and subjectivity analysis, (ii) Document parsing, and (iii) Feature and opinion extraction.

3.1.1 Document Processing and Subjectivity Analysis

We employ *document processing* to divide an unstructured web document into individual record-size chunks, to clean them by removing ML tags, and to present them as individual unstructured record documents for further processing. The cleaned documents are converted into numeric-vectors using unigram model for the purpose of subjectivity analysis. In document vectors a value represents the likelihood of each word being in a subjective or objective sentence.

According to Pang and Lee [9] subjective sentences are expressive of the reviewer's sentiment about the product, and objective sentences do not have any direct or obvious bearing on or support of that sentiment. Therefore, the idea of subjectivity analysis is used to retain segments (sentences) of a review that are more subjective in nature and filter out those that are more objective. This increases the system performance both in terms of *efficiency* and *accuracy*. The idea proposed by Yeh [8]

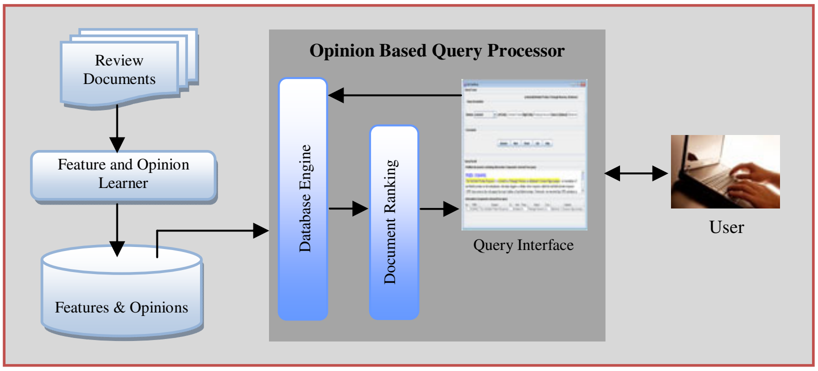


Fig. 1. Proposed opinion-based query answering framework

is used to divide the reviews into subjective parts and objective parts. In [8], the idea of cohesiveness is used to indicate segments of a review that are more subjective in nature versus those that are more objective. We have used a corpus of subjective and objective sentences used in [9] for training purpose. The training set is used to get the probability for each word to be subjective or objective, and the probability of a sentence to be subjective or objective is calculated using the unigram model. The Decision Tree classifier of Weka[1] is trained to classify the unseen review sentences into subjective and objective classes.

3.1.2 Document Parsing

Since our aim is to extract product features and the opinions from text documents, all subjective sentences are parsed using Stanford Parser[2] which assigns Parts-Of-Speech (POS) tags to English words based on the context in which they appear. The POS information is used to locate different types of information of interest inside the text documents. For example, generally noun phrases correspond to product features, adjectives represent opinions, and adverbs are used as modifiers to represent the degree of expressiveness of opinions. Since, it is observed that opinion words and product features are not independent of each other rather, directly or indirectly inter-related through some semantic relations, each sentence is converted into dependency tree using Stanford Parser. The dependency tree, also known as *word-word relationship*, encodes the grammatical relations between every pair of words. A sample POS tagged sentence and the corresponding dependency tree generated using Stanford Parser is shown in Fig. 2(a) and 2(b) respectively.

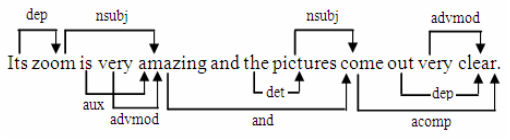
3.1.3 Feature and Opinion Extraction

This process takes the *dependency tree* generated by document parser as input and output feasible information components after analyzing noun phrases and the associated adjectives possibly preceded with adverbs. On observation, we found that

[1] `http://www.cs.waikato.ac.nz/~ml/weka/`
[2] `http://nlp.stanford.edu/software/lex-parser.shtml`

Its/PRP$ zoom/NN is/VBZ very/RB amazing/JJ and/CC the/DT pictures/NNS come/VBP out/IN very/RB clear/JJ ./.

(a) A POS-tagged sentence


dep
nsubj
nsubj
advmod
Its zoom is very amazing and the pictures come out very clear.
aux
advmod
det
and
dep
acomp

(b) Dependency tree

<zoom, very, amazing> // Extracted information component through Rule-1

<pictures, very, clear> // Extracted information component through Rule-2

(c) Extracted Information Components

Fig. 2. (a) A POS-tagged sentence, (b) the corresponding dependency tree generated by Stanford Parser, and (c) extracted information components

product features are generally noun phrases and opinions are either only adjectives or adjectives preceded by adverbs. For example, consider the following opinion sentence:

```
(ROOT(S(NP(NP (DT The) (NN battery) (NN life))(PP (IN of)
(NP (NNP Nokia) (NNP N95))))(VP (VBZ is)(ADJP (RB very)
(JJ good)))(. .)))
```

In the above sentence, "battery life" is a noun phrase and appears as one of the features of Nokia N95 whereas, the adjective word "good" along with the adverb "very" is an opinion to express the concern of reviewer. Therefore, we have defined the information component as a triplet <$\mathcal{F}$, $\mathcal{M}$, $\mathcal{O}$> where, $\mathcal{F}$ is a noun phrase and $\mathcal{O}$ is adjective word possibly representing product feature. $\mathcal{M}$ represents adverb that acts as modifier to represent the degree of expressiveness of $\mathcal{O}$. $\mathcal{M}$ is also used to capture negative opinions explicitly expressed in reviews. The information component extraction mechanism is implemented as a rule-based system which analyzes dependency tree to extract information components. Some sample rules are presented below to highlight the function of the system.

Rule 1: In a dependency tree $\mathcal{T}$, if there exists a *subj*(w_i, w_j) relation such that POS(w_i) = JJ*, POS(w_j) = NN*, w_i and w_j are not stop-words[3] then w_j is assumed to be a *feature* and w_i as an *opinion*. Thereafter, the relation *advmod*(w_i, w_k) relating w_i with some adverbial words w_k is searched. In case of presence of *advmod* relation, the information component is identified as $<w_j, w_k, w_i>$ otherwise $<w_j, \text{-}, w_i>$.

Rule 2: In a dependency tree $\mathcal{T}$, if there exists a *subj*(w_i, w_j) relation such that POS(w_i) = VB*, POS(w_j) = NN*, and w_j is not a stop-word then we search for *acomp*(w_i, w_m) relation. If *acomp* relation exists such that POS(w_m) = JJ* and w_m is not a stop-word then w_j is assumed to be a *feature* and w_m as an *opinion*. Thereafter, the modifier is searched and information component is generated in the same way as in rule 1.

[3] A list of 571 stop-words available at `http://www.aifb.uni-karlsruhe.de/WBS/aho/clustering`

Fig. 2(c) presents two sample information components extracted by applying these rules on the dependency tree shown in figure 2(b). Though a large number of commonly occurring noun and adjective phrases are eliminated due to the design of the information component itself, it is found that further processing is necessary to consolidate the final list of information components and thereby the product features and opinions. During the consolidation process, we take care of two things. In the first stage, since product features are the key noun phrases on which opinions are applied, so a feasible collection of product features is identified using mutual information [12] value calculated using equation (4). In the second stage of analysis, however, for each product feature the list of all opinions and modifiers is compiled that are used later for indexing and query answering purpose.

The mutual information measure, *I(x, y)*, is used to compare the probability of observing *x* and y *together* with the probabilities of observing *x* and *y independently*. If there is a genuine association between *x* and *y*, then the joint probability *P(x, y)* will be much larger than *P(x).P(y),* and consequently *I(x, y)* >> 0. If there is no interesting relationship between *x* and *y*, then $P(x, y) \approx P(x).P(y)$, and thus, $I(x, y) \approx 0$. If *x* and *y* are in complementary distribution, then *P(x, y)* will be much less than *P(x).P(y),* forcing *I(x, y)* << 0. The probabilities *P(x)* and *P(y)* are estimated by counting the number of observations of *x* and *y* in a corpus, *f(x)* and *f(y)*, and normalizing by *N*, the size of the corpus. The joint probabilities, *P(x, y),* are estimated by counting the number of times that *x* is followed by *y* or *y* is followed by *x* in a window of 5 words to consider structural relationship, and normalizing by *N*. In our application, a list of seed opinion words (positives and negatives) is compiled and mutual information value for a feature word with each of them is calculated. If the cumulated sum value for a feature is zero (i.e., the feature is not associated with any seed opinion word) the feature and the corresponding information component is filtered out, otherwise retained. Thereafter, for each retained feature, the list of opinion words and modifiers are compiled from information components and are stored in a structured form. A partial list of product features, opinions, and modifiers extracted from a corpus of 86 customer reviews on *digital camera* (obtained from www.ebay.com) and from 95 review on mp3 player (used in [3]) is shown in table 1.

$$I(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)} \tag{4}$$

Table 1. A partial list of extracted features, opinions and modifiers for digital camera

Product	Feature	Modifier	Opinion
Digital Camera	picture	not, really, very	beautiful, clear, fantastic, good, great, professional, sharp
	battery	Very	decent, excellent, rechargeable, short, long
	price	---	cheap, excellent, good, great, high
mp3 Player	player	Very	awesome, delicate, perfect, fast, good, great, terrific, large, excellent
	Sound	pretty, very, indeed	excellent, good, wonderful, excellent, great, awesome
	Software	very, somewhat, enough	great, easy, nice, awful, good, smooth, quick, decent, inferior, awesome, installed, bad

3.2 Opinion-Based Query Processor

In this section, we present the query processing mechanism using BK-FIRM model over structured repository of features and opinions extracted by Feature and Opinion Learner module. To apply BK-FIRM in our case, *D* is the set of all review documents under consideration; the set T is generated for each product feature and it contains all opinion words associated with a particular feature. Thus, for each feature, a fuzzy relation matrix $\mathfrak{R}_F$ is generated in which contents are normalized *tf-idf* values. In order to handle queries on multiple features a user can use fuzzy logic connectives such as AND, OR and NOT, and fuzzy quantifiers as defined in equations (5) to (9).

$$a\ OR\ b = \mu(a) \vee \mu(b) = \max(\mu(a), \mu(b)) \tag{5}$$

$$a\ AND\ b = \mu(a) \wedge \mu(b) = \min(\mu(a), \mu(b)) \tag{6}$$

$$NOT\ \ a = \neg\mu(a) = 1 - \mu(a) \tag{7}$$

$$VERY\ \ a = Q_{very}(\mu(a)) = [\mu(a)]^2 \tag{8}$$

$$FAIRLY\ \ a = Q_{FAIRLYy}(\mu(a)) = [\mu(a)]^{1/2} \tag{9}$$

To illustrate this process, a partial view of fuzzy relations $\mathfrak{R}_{picture}$ and $\mathfrak{R}_{price}$ for two camera features *picture* and *price* are shown in equations (10) and (11) respectively. Given a query *Q = VERY(sharp) AND FAIRLY(high),* i.e., *camera with very sharp picture quality and fairly high price*, we get a fuzzy set D_F (equation 12) which represents the suitability between documents and the query. When α-cut = 0.7 is applied, we can get the documents d_1 and d_2 in this order of relevance.

$$\mathfrak{R}_{picture} = \begin{array}{c} \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{array} \begin{array}{c} \begin{array}{cccc} beautiful & good & sharp & clear \end{array} \\ \left[\begin{array}{cccc} 0.7 & 0.6 & 0.9 & 1.0 \\ 0.6 & 0.8 & 1.0 & 0.4 \\ 0.5 & 0.7 & 0.8 & 0.7 \\ 0.2 & 0.7 & 0.6 & 0.5 \\ 0.3 & 0.2 & 0.7 & 0.5 \end{array}\right] \end{array} \tag{10}$$

$$\mathfrak{R}_{price} = \begin{array}{c} \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{array} \begin{array}{c} \begin{array}{cccc} cheap & good & great & high \end{array} \\ \left[\begin{array}{cccc} 0.4 & 0.3 & 0.6 & 0.8 \\ 0.6 & 0.5 & 0.7 & 0.5 \\ 0.4 & 0.2 & 0.3 & 0.5 \\ 0.3 & 0.8 & 0.7 & 0.6 \\ 0.9 & 0.4 & 0.5 & 0.3 \end{array}\right] \end{array} \tag{11}$$

$$D_F = \{d_1/0.81, d_2/0.71, d_3/0.64, d_4/0.36, d_5/0.49\} \tag{12}$$

4 Experimental Results

In this section, we present the experimental details of the feature and opinion mining process. For subjectivity analysis, we used the subjectivity dataset[4] v1.0 from Cornell for training purpose. The dataset consists of 5000 subjective sentences and 5000 objective sentences. A Java program is written to extract features using unigram model from this dataset and to convert each sentence into equivalent numeric vector where a value represents likelihood of each word being in a subjective or objective sentence. Thereafter, the Decision Tree classifier of Weka is trained to classify the unseen sentences into subjective and objective classes. The accuracy of the classier on 10-fold cross validation is 82%. The data sample used in our work to mine features and opinions for customer reviews summarization consists of 86 review documents on different models of digital camera (Canon: 60, Panasonic: 26) – all obtained from www.ebay.com, and 95 documents on mp3 player used in [3]. The feature and opinion extraction process described in section 3.1.3 was implemented using Java to mine features and opinionated words along with modifiers from the subjective review sentences. Initially, a total of 48 and 227 for *digital camera* and *mp3 player* respectively were extracted out of which only 33 and 151 were retained after feasibility analysis. For each retained feature, the list of both opinions and modifiers were compiled, a partial view of which is shown in table 1, and stored in structured database. Thereafter, queries were processed over this database using BK-FIRM model to extract relevant review documents.

4.1 Evaluation Methods

The performance of the whole system is analyzed by taking into account the performance of the *feature and opinion extraction* process only as it is difficult to provide a performance analysis of the query-processing module, since no benchmark set of queries are available for judging the performance of such a system. Since the information components are finally stored in a database, the system can obviously retrieve all exact matches correctly. When it comes to judging the relevance of answers to fuzzy query, the quality of retrieval is dependent on the similarity computation procedure. For example, it can be seen from the examples cited above that in some cases, the fuzzy Min-Max function seems to be too restrictive, though we have chosen it since this provides a standard way of interpreting AND and OR boolean operators. We refrain from giving any relevance figure for this module, since acceptability of an answer generated is largely dependent on the user's perspective.

We now present a discussion on the performance of the whole system which is analyzed by taking into account the performance of the *feature* and *opinion* extraction process. Since terminology and complex proper names are not found in Dictionaries, an obvious problem of any automatic method for concept extraction is to provide objective performance evaluation. Therefore manual evaluation has been performed to judge the overall performance of the system. For evaluation of the experimental results, we use standard Information Retrieval performance measures. From the extraction results, we calculate the true positive *TP* (number of correct feature-opinion pairs the system identifies as correct), the false positive *FP* (number of incorrect feature-opinion pairs the system falsely identifies as correct), true negative

[4] `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

TN (number of incorrect feature-opinion pairs the system identifies as incorrect), and the false negatives *FN* (number of correct feature-opinion pairs the system fails to identify as correct). By using these values we calculate the following performance measures:

Precision (π): the ratio of true positives among all retrieved instances.

$$\pi = \frac{TP}{TP + FP} \tag{13}$$

Recall (ρ): the ratio of true positives among all positive instances.

$$\rho = \frac{TP}{TP + FN} \tag{14}$$

F1-measure (F_1): the harmonic mean of recall and precision.

$$F_1 = \frac{2\rho\pi}{\rho + \pi} \tag{15}$$

Accuracy (τ): the ratio of sum of TPs and TNs over total positive and negative instances.

$$\tau = \frac{TP + TN}{TP + FP + FN + TN} \tag{16}$$

The values of the above performance measures are calculated for each category of experimental data. In order to present a synthetic measure of performance over all categories, we present the macro-averaged performance which consists in simply averaging the result obtained on each category. Table 2 summarizes the performance measure values for our system in the form of a misclassification matrix. The recall value is lower than precision indicating that certain correct feature-opinion pairs could not be recognized by the system correctly. This is justified since most of the reviewers do not follow grammatical rules strictly while writing reviews due to which the parser fails to assign correct POS tag and thereby correct dependency relations between words. However, almost all identified feature-concept pairs are correct, which leaves scope for enhancing our grammar to accommodate more dependency relations. After analyzing the review documents manually we also found that some review documents contain junk sentences too which opens a new direction of research – *review spam analysis*.

Table 2. Performance evaluation of feature-opinion extraction process

Product Name		**TP**	**FP**	**FN**	**TN**	**Precision (%)**	**Recall (%)**	**F1-measure (%)**	**Accuracy**
Digital Camera	**Canon**	34	03	26	314	91.89	56.67	70.10	92.30
	Panasonic	31	03	17	174	91.18	64.58	75.61	91.11
mp3 player		264	33	149	1287	88.89	63.92	74.37	89.50
					Macro-Average	**90.65**	**61.72**	**73.36**	**90.97**

5 Conclusion and Future Work

In this paper, we have proposed an opinion-based query answering framework which performs linguistic and semantic analysis of text to identify product features and opinions from review documents. We have also proposed a method using BK-FIRM model to handle imprecise user queries, formulated using fuzzy quantifiers, over review documents. Presently, we are refining the rule-set to consider more dependency relations to improve the *precision* and *recall* values of the system. Instead of directly using standard membership functions for fuzzy quantifiers and ignoring the one present in review documents for relevance computation, we are also exploring a fuzzy similarity computation method that would consider both the quantifiers present in user query and in retrieved documents for relevance computation.

References

1. Kohout, L.J., Keravnou, E., Bandler, W.: Automatic Documentary Information Retrieval by Means of Fuzzy Relational Products. In: Gaines, B.R., Zadeh, L.A., Zimmermann, H.J. (eds.) Fuzzy Sets in Decision Analysis, pp. 308–404 (1984)
2. Ding, X., Liu, B., Philip, S.Y.: A Holistic Lexicon-Based Approach to Opinion Mining. In: Proceedings of the first ACM International Conference on Web search and Data Mining (WSDM'08), California, USA, pp. 231–240 (2008)
3. Hu, M., Liu, B.: Mining and Summarizing Customer Reviews. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04), USA, pp. 168–177 (2004)
4. Liu, B., Hu, M., Cheng, J.: Opinion Observer - Analyzing and comparing opinions on the Web. In: Proceedings of the 14th International Conference on World Wide Web (WWW'05), Japan, pp. 342–351 (2005)
5. Kim, S., Hovy, E.: Determining the Sentiment of Opinions. In: Proceedings of the 20th International Conference on Computational Linguistics (COLING'04), Switzerland, pp. 1367–1373 (2004)
6. Popescu, A.M., Etzioni, O.: Extracting Product Features and Opinions from Reviews. In: Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP'05), Canada, pp. 339–346 (2005)
7. Ghose, A., Ipeirotis, P.G.: Designing Ranking Systems for Consumer Reviews: The Impact of Review Subjectivity on Product Sales and Review Quality. In: Proceedings of the Workshop on Information Technology and Systems (WITS'06), Milwaukee (2006)
8. Yeh, E.: Final Project Picking the Fresh from the Rotten: Quote and Sentiment Extraction from Rotten Tomatoes Movie Reviews, CS224N/Ling237 (2006)
9. Pang, B., Lee, L.: A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In: Proceedings of ACL'04, pp. 271–278 (2004)
10. Turney, P.: Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL'02), Philadelphia, Pennsylvania, pp. 417–424 (2002)
11. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment Classification Using Machine Learning Techniques. In: Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP'02), USA, pp. 79–86 (2002)
12. Church, K.W., Hanks, P.: Word Association Norms, Mutual Information, and Lexicography. Computational Linguistics 16(1), 22–29 (1990)

Blog Opinion Retrieval Based on Topic-Opinion Mixture Model

Peng Jiang[1], Chunxia Zhang[2], Qing Yang[1], and Zhendong Niu[1]

[1] School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China
[2] School of Software, Beijing Institute of Technology, Beijing 100081, China
{jp,cxzhang,yangqing2005,zniu}@bit.edu.cn

Abstract. Recently, as blog is becoming a popular medium to express opinions, blog opinion retrieval excites interest in the field of information retrieval. It helps to find and rank blogs by both topic relevance and opinion relevance. This paper presents our topic-opinion mixture model based approach to blog opinion retrieval in the TREC 2009 blog retrieval task. In our approach, we assume each topic has its own opinion relevance model. A topic-opinion mixture model is introduced to update original query model, and can be regarded as a mixture of topic relevance model and opinion relevance model. By pseudo-relevance feedback method, we can estimate these two models from topic relevance feedback documents and opinion relevance feedback documents respectively. Therefore our approach does not need any annotated data to train. In addition, the global representation model is used to represent an entire blog that contains a number of blog posts. Experimental results on TREC blogs08 collection show the effectiveness of our proposed approach.

Keywords: topic-opinion mixture model, blog, opinion retrieval, rank.

1 Introduction

In resent years, blog is becoming an increasingly popular form of communication on the World Wide Web. The blogosphere is a rich information source of public voice, and is useful in extracting and mining public opinions towards some objects or events. Different from other kinds of online textual information, the main characteristics[1] of a blog are: 1) Information provided is often opinion-oriented; 2) Containing numbers of documents that cover a wide range of topics. The need to find appropriate retrieval techniques to track the way bloggers react to products, persons and events raises some challenging problems in the field of information retrieval[2]. Blog opinion retrieval is a task to save the challenge and serve the growing interest in IR.

In this paper, blog opinion retrieval is defined as a task to search blogs with a recurring interest and opinion towards a given topic. Similar to traditional retrieval system, blog opinion retrieval has two basic tasks: 1) search the relevant documents to a user's query, and 2) ranking these documents according to the level of relevance. However, blog opinion retrieval has several special characteristics to be taken into consideration. The goal of blog opinion retrieval is to find blogs that are principally devoted to certain topics over the time span of the blogs, and to recommend user to subscribe as an interesting feed about the topic (i.e. users may add the interesting feed to their RSS readers). This requires the retrieval unit to be the entire blog containing a number of posts, but not a single post document. Since a blog contains both relevant posts and non-relevant posts to a topic, the overall relevance of a blog must be measured in a proper way. Besides, the blog opinion retrieval goes beyond topic relevance and integrates the opinion relevance in the evaluation of the retrieved blogs. This requires the system to determine whether a blog expresses opinions or facts.

TREC 2009 Blog Track[1] highlights its interest in blog retrieval, and introduces the Faceted Blog Distillation Task. This task takes into account a number of attributes of facets such as opinion, personality and in-depth facets. This paper mainly focuses on the blog retrieval on opinion facet. Technically, there are two typical approaches to the blog opinion retrieval in previous works: two-stage approach based on classification and mixture of language models approach. The two-stage approach is often used in previous TREC Blog Track. There are two basic components in this approach[3]: the retrieval component and the opinion classification component. The former carries out basic relevance retrieval for each query whereas the latter classifies each blog into two categories, namely, opinionated category and factual category. SVM and the maximum entropy classifiers are used in many cases. Mixture of language models approach[4, 5] assumes that a blog is generated by sampling words from a mixture model involving a background language model, a topic language model, and an opinion language model.

In this paper, we present our approach based on the topic-opinion mixture model. It is similar to the above mentioned mixture of language model approach. However, their approaches assume the content of opinion model is the same for all topics, or require models to be trained for every topic by annotated data, or manually input subjective keywords. In our approach, we assume the text opinion expression is dependent on the topic. We first make use of pseudo feedback documents from wiki corpus to construct the topic relevance model, and then some words are automatically selected from a subjective/objective lexicon by the semantic association extent with the topic. Then we combine these words with original query to re-retrieve and get the opinion feedback documents. An opinion relevance model is constructed by these feedback documents. Finally, a topic-opinion mixture model is combined from topic relevance model and opinion relevance model. This model contains topic features and their associated opinion features. So it is effective to evaluate the level of topic relevance and opinion relevance of a blog.

We conduct experiments in this paper on TREC blogs08 datasets, with each blog post being considered as a web page. Moreover, the opinion lexicon (subjective or objective lexicon) used is domain-independent. Hence our proposed approach is applicable to all opinion retrieval tasks on any text resource contained information about topic and opinion, such as product reviews.

The rest of the paper is organized as follows. In Section 2, we briefly introduce the related works in the field. The problem is defined in Section 3. The whole approach is described in Section 4. The experiments and result analysis are presented in Section 5. Finally we conclude the paper and discuss the future work in Section 6.

[1] `http://ir.dcs.gla.ac.uk/wiki/TREC-BLOG`

2 Related Works

There are many related works in the TREC Blog Track. First introduced in TREC 2006, the blog track explores the information seeking behavior in the blogosphere. In the past years, the track had two main tasks: the opinion finding task and the blog distillation task. Normally a two-stage process is used to address the opinion finding task. At the first stage, documents are ranked using modern and effective document ranking functions such as BM25[6], language models (LM) and divergence from randomness (DFR) models[7]. A relevance score is allocated to each document. At the second stage of the retrieval process, the classifier [8-12] is used to determine whether a document is opinionated or factual, and an opinionated score is assigned for the document. Next the retrieved documents are re-ranked according to the combined score of the relevance score and the opinion score. Most solutions use a linear combination of relevance score and opinion score, whereas a quadratic combination solution[13] is proposed and achieve a significant improvement.

For the blog distillation task, there are three main solutions: expert finding, pseudo-cluster selection and federated search model. Expert finding solution[7, 14] regards the blog distillation task as an association finding task, between topics and bloggers. blogger model and posting model are proposed for modeling blog distillation[15]. The blogger model represents the blog as a as a multinomial probability distribution over the vocabulary terms. It then computes probability of a query given a blogger. While in the posting model, each post is computed by query likelihood scoring method followed by combining the score for each post. Pseudo-cluster selection solution[16] samples K relevant posts from a blog, and then virtually combines these posts into a topic-dependent pseudo-cluster. Federated search model solution[17] ranks blogs by the estimated number of relevant documents. Pseudo-cluster selection and federated search model solutions use small document model which treats posts of a blog individually. In expert finding solution, large document model which treats all posts of a blog as a whole can achieve a better performance than the small document model. All solutions use language model as the basic retrieval method.

In TREC 2009 Blog Track, the opinion finding task and the blog distillation task are merged into a new task, called faceted blog distillation. Opinion is one of three facets. This paper mainly focuses on the opinion facet. We use a mixture of topic and opinion language models to solve the problem of blog opinion retrieval. A mixture of language models is commonly used in IR application. The basic idea[18] is to infer language models corresponding to unobserved features in the corpus, with the hope that the features learned represent topic and opinion. An example of these works is from Koji and Victor[5], in which sentiment relevance models and topic relevance models are combined based on Generative Models. Mei ant others[4] first introduced Topic-sentiment Mixture model (TSM), which can reveal the latent topical facets in a blog collection, the subtopics in the results of an ad hoc query, and their associated opinions. Their TSM model is a special case of CPLSA model[19], which mixes themes with different views. TSM attempts to learn a general opinion model to all topics, based on the assumption that the opinion model is independent to the topic model. However, in reality, there is a correlation between opinion model and topic model. For example, in topic “wii exercise”, the words represent opinion such as “magical”, “disgust”, “silly” have a higher probability of occurrence; while in topic "westerns movies and novels", the opinionated words such as "flawless", "oddities", "propitiously" are more likely to appear. Our approach assumes each topic has its own opinion relevance model. The opinion relevance model can be estimated by pseudo-relevance feedback, and then combined with topic relevance model which is estimated by wiki pseudo-relevance feedback.

3 Problem Definition

The aim of opinion blog retrieval task is to "find opinionated or factual blogs that are principally devoted to a given topic[2] over the timespan of the blog". Inspired by TREC 2009 Blog Track, we define the opinion blog retrieval task as follows:

Given a topic *T*, find blogs related to *T*, rank them by topic relevance and opinion relevance. The system should provide three blog ranking results according to opinionated relevance, factual relevance and topic relevance as the baseline respectively. The retrieval unit is a blog containing a number of blog posts which can be viewed as web documents.

The previous solution to blog opinion retrieval problem adopted a two-stage strategy: 1) Topic relevance retrieval that finds all topic relevant blogs, regardless of the opinion relevance; 2) Using different classification techniques to compute the opinion relevance of all retrieved blogs, followed by re-ranking them. In the following section, we introduce our approach based on the topic-opinion mixture model to address the blog opinion retrieval task.

4 Our Approach to Blog Opinion Retrieval

4.1 Blog Representation and Query Generation

Following the works of [17, 20], we choose Global Representation Model to represent blog. This model treats a blog as a virtual document which is composed of all posts of the blog. Because this model considers all posts over the timespan of the blog, it can factually reflect the recurring interest of the blog. In addition, since we use language model based approach to rank, Global Representation Model, which combines many posts into a large document, can avoid the problem of sparsity of words as much as possible.

In our approach, title, description and narrative fields of a topic are used for query string generation. First, we filter out unnecessary punctuation marks in the above fields. All verbs are replaced by their infinitives and all nouns by their singular forms. After this, we extract the keywords to build the bag of words. The basic Indri[3] query Q is defined as:

$$\#combine(w_1\ w_2 \ldots w_n)$$

$w_1\ w_2 \ldots w_n$ are the keywords in the bag. We use the following Indri query template to generate query string for a given topic:

[2] Topic in TREC mainly includes three fields: title, description and narrative.

[3] Indri is a search engine from the Lemur project.
`http://www.lemurproject.org/indri/`

#weight(0.5 Q_{title} 0.3 $Q_{description}$ 0.2 $Q_{narrative}$)

where Q_{title}, $Q_{description}$ and $Q_{narrative}$ are basic Indri queries generated by title, description and narrative field of the topic.

4.2 Basic Retrieval Model

Using the language model approach in IR has shown its effectiveness and simplicity. The general language model approach[21] is decomposed into three components: 1) query model $_Q$; 2) document model $_D$; 3) matching strategy between query model and document model. In our approach, we choose KL-divergence to measure the distance between $_Q$ and $_D$, and rank blogs by the following formula:

$$\begin{aligned} score(D,Q) &= -D(\theta_Q \parallel \theta_D) \\ &= -\sum_w p(w \mid \theta_Q) \log \frac{p(w \mid \theta_Q)}{p(w \mid \theta_D)} \\ &= \sum_w p(w \mid \theta_Q) \log p(w \mid \theta_D) + cons(\theta_Q) \end{aligned} \tag{1}$$

Because the constant *cons*($_Q$) does not affect the ranking results, we do not compute it in our system. Thus, the main task is to estimate $_Q$ and $_D$. For blog retrieval in the paper, the document model $_D$ is a multinomial distribution whose parameters are represented by unigram language models. We assume that blog documents are generated by $_D$, which can be estimated by the following formula:

$$p(w \mid \theta_D) = \frac{c(w,D) + \mu p(w \mid C)}{|D| + \mu} = \frac{\sum_{d \in D} c(w,d) + \mu p(w \mid C)}{\sum_{d \in D} |d| + \mu} \tag{2}$$

where $p(w|C)$ is a background language model, d is a post of blog D, $c(w,d)$ is the count of w occurs in d, and μ is a Dirichlet smoothing parameter. We use μ=2000 in this paper, which is optimal in most cases[22].

In traditional approach[21], $_Q$ will be updated by feedback documents model that can be obtained by the relevant documents judged by users, or top documents from initial retrieval. To address the special need for blog opinion retrieval, we introduce Topic-opinion Mixture model $_{TO}$, and interpolate it with the original query model $_Q$ to obtain the updated query model $_{Q'}$, and then assign a score to blog D by Formula (1). The updated query model $_{Q'}$ is:

$$\theta_{Q'} = (1-\alpha)\theta_Q + \alpha\theta_{TO} \tag{3}$$

where α controls the influence of topic-opinion mixture model $_{TO}$. In Section 4.3, we describe how to estimate topic-opinion mixture model $_{TO}$.

4.3 Topic-Opinion Mixture Model

The topic-opinion mixture model $_{TO}$ in Formula (3) is the language model which reflects the information need for both topic and opinion; hence a mixture of language models is used to estimate $_{TO}$. In our solution, we define two language models, namely, topic relevance model $_{T}$ and opinion relevance model $_{O}$. The topic-opinion mixture model $_{TO}$ is a linear combination of the two language models:

$$\theta_{TO} = (1-\beta)\theta_T + \beta\theta_O \tag{4}$$

where β is used to control influence of opinion relevance model $_{O}$.

In general, the topic relevance model $_{T}$ in Formula (4) can be obtained by pseudo-relevance feedback method (PRF). PRF assumes the k top-retrieved documents are relevant to the original query and extracts highly discriminative words from those documents to update the original query model. We use divergence minimization algorithm[21] to estimate $_{T}$. The divergence minimization algorithm assumes that the topic relevance model is very close to each language model of feedback documents, and uses KL-divergence as the distance between two language models. In order to obtain the feedback documents with high relevance, we index the Wikipedia corpus[4] and treat the k top-retrieved wiki pages as the relevance feedback documents. Given a topic T, let F= $\{d_1, \ldots d_k\}$ be a set of top k retrieved feedback documents from Wikipedia corpus. So the distance can be represented as:

$$D(\theta_T, F) = \frac{1}{k}\sum_{i=1}^{k} D(\theta_T \parallel \theta_{d_i}) - \lambda D(\theta_T \parallel \theta_{Wiki}) \tag{5}$$

Where $_{Wiki}$ is the Wikipedia corpus language model, $\lambda \in [0, 1)$ is the factor that controls the weight of Wikipedia corpus language model. Following [21], $p(w|_{T})$ can be computed as follows:

$$p(w \mid \theta_T) \propto \exp\left(\frac{1}{1-\lambda}\frac{1}{k}\sum_{i=1}^{k}\log p(w \mid \theta_{d_i}) - \frac{1}{1-\lambda}\log p(w \mid \theta_{Wiki})\right) \tag{6}$$

According to Formula (6), words that are common in the feedback documents, but not common in the entire Wiki corpus will be assigned a higher probability. In our system, k=25, λ=0.5, the feedback terms count is set to be 100.

Next we must estimate the opinion relevance model $_{O}$ in Formula (4). $_{O}$ reflects the users' information need for opinion. Some bloggers provide opinionated content for their interested topics, while others report factual information. So we need to estimate two $_{O}$, one for opinionated information and the other for factual information. Previous works show that the opinion always has an association with topic. Different topics may have a different opinion expression. But training different models on annotated data for different topic is usually unpractical.

The basic procedure of our approach has two steps. The first step is to expand original query with some subjective words or objective words, and then use the expanded query to obtain the top k ranked results as pseudo-feedback documents. The second step is to make use of pseudo-relevance feedback method to estimate $_{O}$. For the first step, the most important thing is to select m subjective/objective words that have the closest association with a given topic. In our solution, we use a subjective lexicon and an objective lexicon. The subjective lexicon contains 8821 words that are

[4] `http://download.wikimedia.org/enwiki/`

used in OpinionFinder[23]. The words in objective lexicon are selected from SentiWordNet[24]. Similar to [25], we use the Pointwise Mutual Information (PMI) to measure the semantic association between subjective/objective word *w* and the query string *Q* of a given topic:

$$PMI\left(w,Q\right)=\log\frac{p\left(w,Q\right)}{p\left(w\right)p\left(Q\right)}=\log\frac{hits\left(\#uw15\left(w\quad Q\right)\right)\times\left|C\right|}{hits\left(w\right)\times hits\left(Q\right)} \tag{7}$$

where |*C*| is the total number of documents in corpus. We make use of blog collection index to estimate PMI. *hits*(*w*) and *hits*(*Q*) are the counts of retrieved documents which contain subjective/objective word *w* and query string *Q* respectively. *hits*(*#uw*15(*w* *Q*)) is the count of retrieved documents containing *w* and *Q* simultaneously in an unordered window of 15 terms. The reason why we use a fixed size window instead of a sentence is that: it is time-consuming and unpractical to split all text into sentences, and the inaccuracy can be ignored when large corpus is used. To avoid division by zero, 0.01 is added to the number of hits. Finally we choose the top 30 subjective/objective words according to the PMI value, and use them to expand original query. The feedback documents can be used to build opinion relevance model $_{O}$ by Formula (6).

5 Experiments

5.1 Experiment Setup

5.1.1 Data Sets

We use TREC Blogs08 collection as required by TREC 2009 Blog Track to evaluate our approach. The summary statistics of this collection is shown in Table 1. We actually use the permalinks and homepages in our approach. Blog feeds collection is not used. It is because the text in the feed pages usually contains a few sentences of each post and therefore cannot reflect the topic or opinion well. The permalinks and homepages are encoded by HTML. We use Indri to index them respectively. The Krovetz stemmer and a list with 450 stop words are used to pre-process.

Table 1. Summary statistics of data sets

Data Set	Doc number	Size (Uncompressed)	Time span
homepages	1,011,733	56G	14/01/2008
feeds	1,303,520	808G	~
permalinks	28,488,767	1445G	10/02/2009

5.1.2 Evaluation

There are 13 opinion topics provided by TREC 2009 Blog Track (see Table 2). The evaluation metrics used are standard IR measures[26], such as mean average precision (MAP), R-Precision (R-prec), and precision at top 10 results (p@10). The relevance and opinion judgments adopt the TREC 2009 Blog Track standards: not judged (-1), not relevant (0), relevant (1), relevant and opinionated (2) and relevant and factual (3). All results are assessed by the evaluation tool provided by TREC.

There are four approaches in our experiments for comparative studies: (1) Our Topic-opinion Mixture Model (TOM) (2) MEClassifier. It is a traditional approach based on classifier. We trained a maximum entropy classifier on Movie Review Data. The classifier takes blog text vector as input, and outputs opinionated or factual label and an associated score, which is combined with original relevance score. Blogs is then re-ranked by the combined score. (3) SingleModel. It combines all topic models with the same opinion model. This approach is introduced in [4], which treats the opinion model the same for all topics in a collection. (4) Baseline. It only considers the topic relevance score while ranking the opinionated and the actual blogs.

Table 2. Opinion topics in TREC Blog 2009

No.	Title	No.	Title	No.	Title
1103	farm subsidies	1125	cosmetic surgery	1141	sciatica remedies
1106	taiwan politics	1132	gun control dc	1144	future of journalism
1111	jazz music	1134	new orleans after katrina	1150	NASA space program
1116	homeopathic medicine	1137	civil unions		
1119	no child left behind	1140	scientology		

5.2 Experimental Results

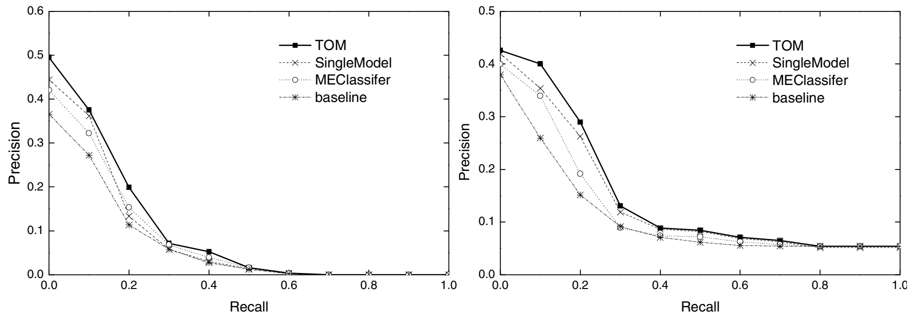
5.2.1 Overview of Experimental Results

Result comparisons of each approach are presented in Table 3 and Fig.1. The results show that all approaches outperform the baseline. Comparing with other approach, our approach achieves the best retrieval performances except for R-prec and P@10 of factual blog retrieval in Table 3. This demonstrates that our proposed approach is effective especially for opinionated blog retrieval.

Fig. 2 (a) and (b) show the performance improvements over baseline on each topic in terms of MAP and R-prec. The average improvements on all topics for opinionated blogs retrieval are 48.87% and 26.39% in terms of MAP and R-prec. The average improvements for factual blogs retrieval are 22.69% and 8.82% in terms of MAP and R-prec. We note that there is a slight improvement over baseline in factual blog retrieval. The explanation is that, ranking by topic and factual relevance does not have much difference from ranking only by topic relevance. Only topic 1134 and 1150 get decreased performance. In terms of MAP, there are 5 topics which have no improvement over baseline for factual blogs retrieval, comparing with 2 topics for opinion blogs retrieval. In terms of R-prec, there are 7 topics which have no improvement over baseline for factual blogs retrieval, comparing with 5 topics for opinion blogs retrieval. This proves that our approach is more effective for opinionated blogs retrieval than factual blogs retrieval.

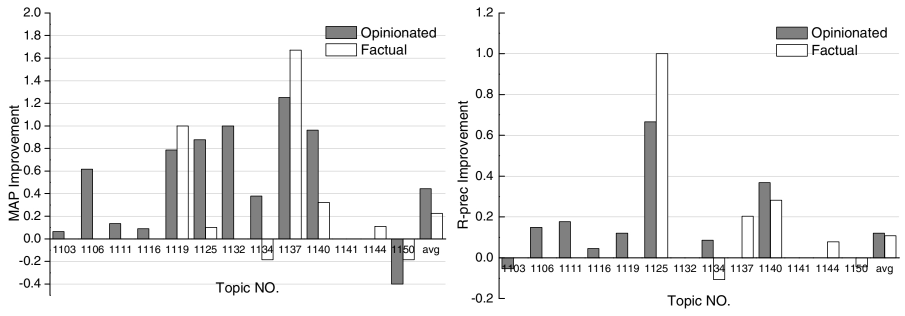
Table 3. Performance comparison among different approaches

Approaches	MAP		R-prec		P@10	
	opinionated	factual	opinionated	factual	opinionated	factual
Baseline	0.0573	0.1124	0.1027	0.1270	0.0923	0.0846
MEClassifer	0.0693	0.1236	0.1298	**0.1402**	0.1000	0.1077
SingleModel	0.0732	0.1159	0.1302	0.1305	0.1154	**0.1231**
TOM	**0.0853**	**0.1379**	**0.1317**	0.1382	**0.1231**	0.1154



(a) Opinionated blogs retrieval (b) Factual blogs retrieval

Fig. 1. Comparison of recall-precision curves among different approaches



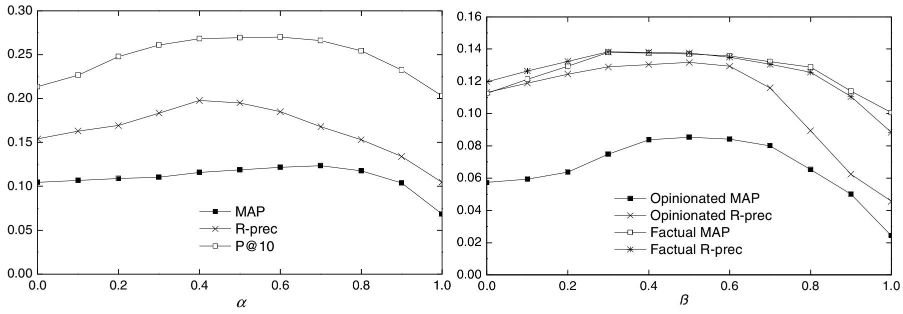
(a) MAP improvement over baseline (b) R-prec improvement over baseline

Fig. 2. Performance improvements over baseline on each topic

5.2.2 Analysis of Parameters of Topic-Opinion Mixture Model

In our approach, the parameter β of the topic-opinion mixture model controls influence of opinion relevance model $_O$. Specifically, β is used to adjust the ratio of topic relevance and opinion relevance in topic-opinion mixture model. In order to analyze the effect of β, we note that parameter α in Formula (3) may affect the final performance. The difference can be observed in Fig. 3 (a), in which we show the changing performances by changing α from 0 to 1, with a step up size of 0.1. In this experiment, we set β=0, thus, $_{TO}$ actually becomes the topic relevance model $_T$. Therefore the experiment actually evaluates the effects of feedback documents from Wiki corpus. We notice that using feedback model from wiki documents can generally improve the performance. But when it is too large approaching 1, the performance is extremely bad and is even worse than the performance without using feedback model. We choose α=0.5, which is a value that can usually achieve better performance than other values.

Fig. 3 (b) shows how MAP, R-prec varies accordingly with β, when α is fixed at 0.5. Note that performance at β=0 is actually the baseline performance. Overall, when the β value increases, the overall performance improves. But when β is too large, the overall performance deteriorates sharply. Be more specific, when β=0.5 the opinionated blog retrieval achieves its best performance; when β=0.3 the factual blog retrieval achieves its best performance. This is because the topic relevance model helps to focus on the topic, while the opinion relevance model can supplement subjective or objective words for the purpose of opinion retrieval. When β is too large, there will be many opinionated or factual blogs with no topic relevance.



(a) Performance sensitivity to α (b) Performance sensitivity to β

Fig. 3. Performance sensitivity to parameters

5.2.3 Analysis of Samples from Topic-Opinion Mixture Model

Table 4 presents sample probabilities using topic-opinion mixture model. Samples are divided into the two topics: "jazz music" and "no child left behind". The "Topic model" columns contain the topic words. These words may come from the subtopic of the corresponding topic, such as "musician", "band", "Africa", "educate", "fund", etc. So they can be treated as supplement for the original query. The "Opinionated model" columns contain subjective words related to the corresponding topic. As we have discussed above, the opinionated relevance model varies significantly with topics. For instance, for "jazz music" topic, the subjective words "limitless", "entertaining" have relatively higher probability of occurrence; whereas for "no child left behind" topic, the associated subjective words are "willing", "supportive", etc. In the "Factual model" columns, the words are found to be neutral, without any semantic orientation. Some words appear in many topics, such as "comment", "state", etc. This reflects that the factual relevance model has low association with topics.

Table 4. Sample probabilities from topic-opinion mixture model. The top 10 words with high probability of occurrence are selected. Results of two topics are presented corresponding to the three language models: topic relevance model, opinionated model and factual model.

Topic 1111 jazz music						Topic 1119 no child left behind					
Topic model		Opinionated model		Factual model		Topic model		Opinionated model		Factual model	
w	$p(w\|\theta_T)$	w	$p(w\|\theta_O)$	w	$p(w\|\theta_O)$	w	$p(w\|\theta_T)$	w	$p(w\|\theta_O)$	w	$p(w\|\theta_O)$
jazz	0.0730	exclusive	0.0137	comment	0.0141	school	0.0343	willing	0.0036	comment	0.0194
music	0.0303	like	0.0137	new	0.0119	student	0.0313	rightly	0.0035	learn	0.0183
play	0.0163	inestimably	0.0040	clear	0.0041	state	0.0263	supportive	0.0035	state	0.0062
musician	0.0148	limitless	0.0040	state	0.0040	nclb	0.0258	benefit	0.0035	question	0.0057
style	0.0133	entertaining	0.0026	profile	0.0039	educate	0.0223	clearly	0.0035	address	0.0052
blue	0.0119	goodly	0.0023	concert	0.0038	fund	0.0219	contentment	0.0035	break	0.0048
new	0.0119	friendly	0.0020	old	0.0037	federal	0.0119	important	0.0035	require	0.0047
band	0.0111	willing	0.0017	live	0.0037	assess	0.0104	transparent	0.0034	public	0.0040
america	0.0107	great	0.0015	swing	0.0032	child	0.0074	winnable	0.0034	legal	0.0039
africa	0.0100	creative	0.0013	classic	0.0031	support	0.0070	justly	0.0027	educational	0.0038

6 Conclusions

In this paper, we present an approach to the task of blog opinion retrieval. This approach uses topic-opinion mixture model to solve the problem of ranking blog not only by topic relevance but also by opinion relevance. Comparing with previous work, this model can effectively learn opinion relevance model without training on annotated data. In addition, the opinion relevance models vary with topics so that the model's effectiveness to different topics is ensured. We evaluate our model on TREC Blogs08 collection, and the experimental results show that the topic-opinion mixture model approach achieves a better performance than other approaches for most of the opinion topics in TREC 2009 Blog Track.

In general, performance of the blog opinion retrieval is worse than traditional text retrieval. There is still a huge potential space for further research to improve the performance of blog opinion retrieval. In addition, it would be interesting to explore the knowledge behind topic and opinion from the perspective of time dimension of blogs. Another interesting future research direction is to use the mixture language model to explore the other blog attributes or facets such as writing style, authority, etc.

Acknowledgments. We would like to express our gratitude to Computer Software Lab of Beijing Institute of Technology for providing us experiment condition for our participation in TREC 09. This work is supported by the grant from Chinese National Natural Science Foundation (No: 60705022).

References

1. Marti, A.H., Matthew, H., Susan, T.D.: What should blog search look like? In: Proceeding of the 2008 ACM workshop on Search in social media. ACM, Napa Valley (2008)
2. Ounis, I., Macdonald, C., Soboroff, I.: On the TREC blog track. In: Proceedings of the International Conference on Weblogs and Social Media (ICWSM), Seattle, USA (2008)
3. Liu, B.: Sentiment Analysis and Subjectivity. CRC Press, Taylor and Francis Group (2009)
4. Qiaozhu, M., Xu, L., Matthew, W., Hang, S., ChengXiang, Z.: Topic sentiment mixture: modeling facets and opinions in weblogs. In: Proceedings of the 16th international conference on World Wide Web, pp. 171–180. ACM, Banff (2007)
5. Koji, E., Victor, L.: Sentiment retrieval using generative models. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, pp. 345–354. Association for Computational Linguistics, Sydney (2006)
6. Lee, Y., Na, S.H., Kim, J., Nam, S.H., Jung, H.Y., Lee, J.H.: Kle at trec 2008 blog track: Blog post and feed retrieval. In: Proceedings of TREC-08 (2008)
7. He, B., Macdonald, C., Ounis, I., Peng, J., Santos, R.L.T.: University of glasgow at trec 2008: Experiments in blog, enterprise, and relevance feedback tracks with terrier. In: Proceedings of TREC-08 (2008)
8. Bermingham, A., Smeaton, A., Foster, J., Hogan, D.: DCU at the TREC 2008 Blog Track. In: Proceedings of TREC-08 (2008)
9. Hoang, L., Lee, S.W., Hong, G., Lee, J.Y., Rim, H.C.: A Hybrid Method for Opinion Finding Task (KUNLP at TREC 2008 Blog Track). In: Proceedings of TREC-08 (2008)

10. Jia, L., Yu, C., Zhang, W.: UIC at TREC 2008 blog track. In: Proceedings of TREC-08 (2008)
11. Li, B., Liu, F., Liu, Y.: UTDallas at TREC 2008 Blog Track. In: Proceedings of TREC-08 (2008)
12. He, H., Chen, B., Du, L., Li, S., Gao, H., Xu, W., Guo, J.: PRIS in TREC 2008 Blog Track. In: Proceedings of TREC-08 (2008)
13. Min, Z., Xingyao, Y.: A generation model to unify topic relevance and lexicon-based sentiment for opinion retrieval. In: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, pp. 411–418. ACM, Singapore (2008)
14. Weerkamp, W., Rijke, M.d.: External Query Expansion in the Blogosphere. In: Proceedings of TREC-08 (2008)
15. Balog, K., de Rijke, M., Weerkamp, W.: Bloggers as experts. In: 31st Annual International ACM SIGIR Conference (SIGIR 2008), pp. 753–754. ACM, Singapore (2008)
16. Seo, J., Croft, W.B.: UMass at TREC 2008 Blog Distillation Task. In: Proceedings of TREC-08 (2008)
17. Jonathan, L.E., Jaime, A., Jamie, C., Jaime, G.C.: Retrieval and feedback models for blog feed search. In: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, pp. 347–354. ACM, Singapore (2008)
18. Bo, P., Lillian, L.: Opinion Mining and Sentiment Analysis. Found. Trends Inf. Retr. 2, 1–135 (2008)
19. Qiaozhu, M., ChengXiang, Z.: A mixture model for contextual text mining. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 649–655. ACM, Philadelphia (2006)
20. Jangwon, S., Croft, W.B.: Blog site search using resource selection. In: Proceeding of the 17th ACM conference on Information and knowledge management, pp. 1053–1062. ACM, Napa Valley (2008)
21. Chengxiang, Z., John, L.: Model-based feedback in the language modeling approach to information retrieval. In: Proceedings of the tenth international conference on Information and knowledge management. ACM, Atlanta (2001)
22. Chengxiang, Z., John, L.: A study of smoothing methods for language models applied to information retrieval. ACM Trans. Inf. Syst. 22, 179–214 (2004)
23. Theresa, W., Janyce, W., Paul, H.: Recognizing contextual polarity in phrase-level sentiment analysis. In: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, pp. 347–354. Association for Computational Linguistics, Vancouver (2005)
24. Esuli, A., Sebastiani, F.: SentiWordNet: A publicly available lexical resource for opinion mining. In: Proceedings of LREC, pp. 417–422 (2006)
25. Turney, P.D., Michael, L.L.: Measuring praise and criticism: Inference of semantic orientation from association. ACM Trans. Inf. Syst. 21, 315–346 (2003)
26. Chris, B., Ellen, M.V.: Retrieval evaluation with incomplete information. In: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 25–32. ACM, Sheffield (2004)

Feature Subsumption for Sentiment Classification in Multiple Languages

Zhongwu Zhai, Hua Xu, Jun Li, and Peifa Jia

State Key Laboratory on Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
CS&T Department, Tsinghua University, Beijing 100084, China P.R.
{zhaizhongwu,junli.cn}@gmail.com

Abstract. An open problem in machine learning-based sentiment classification is how to extract complex features that outperform simple features; figuring out which types of features are most valuable is another. Most of the studies focus primarily on character or word *Ngrams* features, but *substring-group* features have never been considered in sentiment classification area before. In this study, the substring-group features are *extracted* and *selected* for sentiment classification by means of *transductive* learning-based algorithm. To demonstrate generality, experiments have been conducted on *three* open datasets in *three different languages*: Chinese, English and Spanish. The experimental results show that the proposed algorithm's performance is usually superior to the best performance in related work, and the proposed feature subsumption algorithm for sentiment classification is *multilingual.* Compared to the *inductive* learning-based algorithm, the experimental results also illustrate that the *transductive* learning-based algorithm can significantly improve the performance of sentiment classification. As for term weighting, the experiments show that the "tfidf-c" outperforms all other term weighting approaches in the proposed algorithm.

Keywords: Sentiment, Transductive, Substring-group, Multilingual.

1 Introduction

With the growing availability and popularity of online user-generated information, including reviews, forum discussions, and blogs, sentiment analysis and opinion mining ("sentiment analysis" and "opinion mining" denote the same field of study [1]) have become one of the key technologies for handling and analyzing the text data from internet. One of the most widely-studied sub-problems of opinion mining is sentiment classification, which classifies evaluative documents, sentences or words as positive or negative (in some cases, the neutral class is used as well) [2] to help people automatically identify the viewpoints underlying the online user-generated information [3]. Since sentiment classification concerns the opinion expressed in a text rather than its topic, it challenges data-driven methods and resists conventional text classification techniques [4].

Up to this date, machine learning-based methods have been commonly adopted for sentiment classification due to their outstanding performance [3, 4]. An open problem in machine learning-based sentiment classification is how to extract complex features that outperform simple features; figuring out which types of features are most valuable is another [5]. Most of the existing research focus on simple features, including single words [6], character Ngrams [7, 8], word Ngrams [3, 4, 8], phrases [5] and the combination of above features, but *substring-group* features have never been considered in sentiment classification. In fact, the substring-group features based classification approaches have at least the following potential advantages [9], allowing sub-word features and super-word features to be exploited automatically. With such approaches, the messy and rather artificial problem of defining word boundaries in some Asian languages can be avoided, and non-alphabetical features can be taken into account. Furthermore, different types of documents can be dealt with in a uniform way.

In this study, the substring-group features are extracted and selected for sentiment classification by means of the *transductive learning*-based algorithm. **Firstly**, the substring-group features are extracted from the suffix tree constructed by the training and unlabeled test documents, based on the *transductive learning* theory [10]. Since the *substring-groups* include several continuous words or even sentences, the substring-group features facilitate the incorporation of word sequence information to sentiment classification. Also, since the suffix tree is constructed by both training documents and unlabeled test documents, the *structural information of unlabeled test documents* is incorporated to feature extraction. **Secondly**, the extracted substring-group features are further selected to eliminate the redundancy among them. At last, SVM is adopted to classify the unlabeled test documents based on the selected features. Experiments have been conducted on three open datasets in three different languages, Chinese, English and Spanish, and the experimental results demonstrate the effectiveness of the proposed algorithm.

The rest of this paper is organized as follows. Section 2 reviews the learning paradigms and the related work. The proposed algorithm is described in detail in Section 3. The experimental setup is illustrated in Section 4 and the results are given and analyzed in Section 5. Finally, this paper is summarized in Section 6.

2 Learning Paradigms and Related Work

Learning paradigms: Given an example x and a class label y, the standard statistical classification task is to assign a probability, $Pr(y|x)$, to x of belonging to class y. In sentiment classification, the labels are $Y \in \{$'*positive*', '*negative*'$\}$. The data for the sentiment classification task consists of two disjoint subsets: the training set $(X_{train}, Y_{train}) = \{(x_1, y_1), \cdots, (x_N, y_N)\}$, available to the model for its training, and the test set $X_{test} = (x_1, \cdots, x_M)$, upon which we want to leverage the trained classifier to make predictions.

In the paradigm of ***inductive learning***, (X_{train}, Y_{train}) are known, while both X_{test} and Y_{test} are completely hidden during training time. In the case of semi-supervised inductive learning [10-12], the learner is also provided with auxiliary unlabeled data $X_{auxiliary}$, that is not part of the test set. Another setting that is closely related to semi-supervised learning is ***transductive learning*** [10, 13, 14], in which X_{test} (but, importantly, not Y_{test}), is known at training time. One can think of transductive learning as a special case of semi-supervised learning in which $X_{auxiliary} = X_{test}$.

Related work: Sentiment classification can be performed on word level, sentence level and document level. In this paper, we focus on document sentiment classification. Previous studies for sentiment classification on document level can be generally classified into two categories, unsupervised approaches and supervised approaches.

The unsupervised approaches focus on identifying semantic orientation of individual words or phrases, and then classifying each document in terms of the number of these words or phrases contained in each document. Turney determines semantic orientation by phrase Pointwise Mutual Information (PMI) based on pre-defined seed words [15] and rates reviews as thumbs up or down [16]. Kim and Hovy [17] build three models to assign a sentiment category to a given sentence by combining the individual sentiments of sentiment-bearing words. Liu et al. classify customer reviews using a holistic lexicon [18, 19]. Kennedy and Inkpen determine the sentiment of customer reviews by counting positive and negative terms and taking into account contextual valence shifters, such as negations and intensifiers [20]. Devitt and Ahmad explore a computable metric of positive or negative polarity in financial news text [21]. Wan uses bilingual knowledge and ensemble techniques for unsupervised Chinese sentiment analysis [22].

The supervised approaches focus on training a sentiment classifier using labeled corpus. Since the work of Pang et al. [4], various classification models and linguistic features have been proposed. Dave et al. use machine learning based methods to classify reviews on several kinds of products [23]. Pang and Lee report 86.4% accuracy rate of sentiment classification of movie reviews by using word unigrams features for SVMs [3]. Mullen and Collier also employ SVMs to bring together diverse sources of potentially pertinent information, including several favorability measures for phrases and adjectives and knowledge of the topic of the text [24]. Most recently, Li and Sun compare the performance of four machine learning methods for sentiment classification of Chinese reviews using Ngrams features [8]. Blitzer et al. investigate domain adaptation for sentiment classifier [25]. Songbo et al. combine learn-based and lexicon-based techniques for sentiment detection without using labeled examples [26].

To the best of our knowledge, though substring-group features have been used for topic, authorship and genre classification [9], they have not yet been considered in sentiment classification. Moreover, the structural information of *unlabeled* test documents is used in this paper by *transductive learning*, which has not been studied in any related work of sentiment classification. Furthermore, the synergetic effect of feature extracting and feature selecting has been reflected in this study.

3 The Proposed Algorithm

According to the conclusions from the learning paradigms in subsection **2.1**, not only the training documents, but also the unlabeled test documents can be used at training time by the *transductive* learning-based algorithm [10]. The proposed algorithm takes the training documents (*both text and class labels*) and the unlabeled test documents (*only text*) as input, and outputs the predicted classifications of the unlabeled test documents. The framework of the proposed algorithm is shown in Figure **1**, including four stages: *substring-group feature extracting*, term weighting, *feature selecting* and classifying.

3.1 Substring-Group Feature Extracting

The unique substring-group features are extracted by the following steps.

Step (***a***) aims to construct a suffix tree using all the strings of both *training* documents and *unlabeled test* documents. The suffix tree is constructed by Ukkonen's algorithm with *O(n)* time complexity, where n is the number of characters in the text corpus [27]. This step shows the incorporation of *transductive learning*.

In step (***b***), the key-nodes are extracted from the constructed suffix tree. For an *m-character* text corpus, the constructed suffix tree has *m* leaf-nodes and at most *m-1* internal nodes [28]. The text corpus's length *m* is usually a very large number, so it's necessary to extract the key-nodes from the (*2m-1*) nodes. The key-node extracting criteria proposed in [9] is used in this paper, and the recommended values are adopted: L=20, H=8000, B=8, P=0.8 and Q=0.8. The meanings of the parameters are listed in Table 1.

In step (***c***), every suffix of each document is matched with the suffix tree, and all the IDs of the matched key-nodes are taken as the content of the corresponding document.

In step (*d*), from the ***training*** part of the converted documents, all the unique key-node IDs are extracted as features for sentiment classification. This step guarantees that the evaluation of the following experiments is *open test*.

In step (***e***), all the converted documents are translated into corresponding vectors using the unique feature table produced by step (*d*).

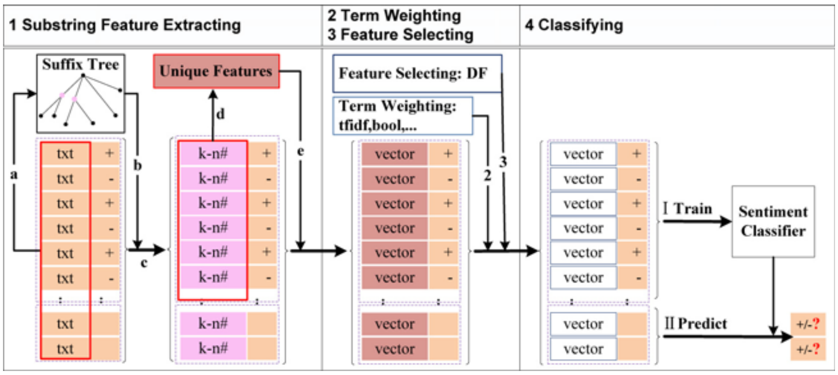


Fig. 1. The framework of the proposed algorithm

According to the definition of suffix tree [28], each node of the suffix tree represents a substring-group of the text corpus. Therefore, the extracted key-node IDs are also called substring-group features in this paper. Moreover, the time complexity of the computing steps a, b and c is linear, which has been proved in [9].

Table 1. The key-nodes extracting parameters

L	The minimum frequency. A node is not extracted, if it has less than L leaf-nodes in the suffix tree.
H	The maximum frequency. A node is not extracted, if it has more than H leaf-nodes in the suffix tree.
B	The minimum number of children. A node is not extracted, if it has less than B children.
P	The maximum parent-child conditional probability. A node u is not extracted, if the probability Pr(v|u) = freq(v)/freq(u) ≥ P, where u is the parent node of v.
Q	The maximum suffix-link conditional probability. A node s(v) is not extracted, if the probability Pr(v|s(v)) = freq(v)/freq(s(v)) ≥ Q, where the suffix-link of v points to s(v).

3.2 Term Weighting

Term frequency has traditionally been used in the standard text classification, but Pang et al. [4] obtained better performance by using presence rather than frequency. Consequently, both term presence ("bool", "three") and term frequency ("tf" and "tfidf-c") are used in this paper. The "tfidf-c" is the variants of standard "tfidf", and it is widely used in text classification [29, 30]. The four adopted term weighting approaches are defined as formulas 1, 2, 3, and 4.

$$bool: \quad \begin{cases} 1 & if\ tf(t_k, d_j) > 0 \\ 0 & if\ tf(t_k, d_j) = 0 \end{cases} \tag{1}$$

$$three: \quad \begin{cases} 2 & if\ tf(t_k, d_j) > 1 \\ 1 & if\ tf(t_k, d_j) = 1 \\ 0 & if\ tf(t_k, d_j) = 0 \end{cases} \tag{2}$$

$$tf: \quad tf(t_k, d_j) \tag{3}$$

$$tfidf\text{-}c: \quad \frac{tf(t_k, d_j) \times log\frac{N}{df(t_k)}}{\sqrt{\sum_{t \in d_j}\left(tf(t_k, d_j) \times log\frac{N}{df(t_k)}\right)^2}} \tag{4}$$

Here, t_k denotes a distinct term corresponding to a single feature; $tf(t_k, d_j)$ represents the number of times term t_k occurs in the document d_j; $df(t_k)$ is the number of documents the term t_k occurs in; N is the total number of training documents.

3.3 Feature Selecting

Document frequency (DF) is the number of documents in which a term occurs. It is the simplest criterion for feature selection and can be easily scaled to a large dataset with linear computation complexity. It is a simple but effective feature selection method for text categorization [31]. In this study, DF is used to pick out the discriminating substring-group features for training and classification.

For DF (Document Frequency) calculation, we compute the document frequency for each feature in the training corpus and then select the top N features with the highest scores. The basic assumption is that the rare features are either non-informative for class prediction, or not influential in global performance.

3.4 Classifying

In this step, the sentiment classifier is trained by machine learning algorithms to predict the classifications of the unlabeled test documents. Due to SVMs' outstanding performance [3, 4, 6, 8, 24, 32], SVMs are adopted in this paper. The SVM^{light} package is used for training and testing with default parameters.

4 Experimental Setup

4.1 Datasets

The proposed algorithm has been tested on three open datasets in three different languages: *Chinese*, *English* and *Spanish*. Table 2 gives a short summary of these open datasets.

Table 2. The summary of the open datasets

Language	Positive	Negative	n-fold CV	Encoding
Chinese_16000[1]	8000	8000	4	GB2312
English_1400[2]	700	700	3	ASCII
Spanish_400[3]	200	200	3	ISO-8859-2

These 160,000 *Chinese* hotel reviews were crawled from the website `http://www.ctrip.com/`, which is one of the most well-known websites in China for hotel and flight reservation. The "*English*_1400" is most commonly used for sentiment classification in English. The *Spanish* corpus is a collection of 400 reviews on cars, hotels, washing machines, books, cell phones, music, computers, and movies. Each category contains 50 positive and 50 negative reviews, defined as positive or negative based on the number of stars given by the reviewers.

In order to compare the results from the related works on these open datasets, *4-fold*, *3-fold* and *3-fold* **cross validation** are used respectively in the following experiments.

4.2 Evaluation Metrics

To evaluate the performance of the proposed algorithm for sentiment classification, we adopted traditional evaluation metric *accuracy* that is generally used in text categorization [30]. In addition, *microF1* and *macroPrecision* are also computed to compare with the related work.

[1] `http://nlp.csai.tsinghua.edu.cn/~lj/pmwiki/index.php?n=Main.DataSet`

[2] `http://www.cs.cornell.edu/people/pabo/movie-review-data/mix20_rand700_tokens.zip`

[3] `http://www.sfu.ca/~mtaboada/research/SFU_Review_Corpus.html`

5 Experimental Results

5.1 Comparisons

To compare to the algorithms in related work, the best performances of the existing typical methods on each dataset are listed in Table 3, respectively. The column "#Features" is the number of features when the best performance is achieved.

Table 3. Comparisons with the best performance of the existing typical methods

Language	Techniques	Best Performance(%)	#Features
Chinese (16,000)	SVM(word bigrams, tfidf-c) [8]	91.2^{microF1}	251,289
	SVM(character bigrams, tfidf-c) [8]	91.6^{microF1}	128,049
	SVM(*key substring-groups* + DF, tfidf-c)	$\mathbf{94.0}^{\text{microF1}}$	41,454
English (1,400)	SVM(character unigrams, bool) [4]	82.9^{accuracy}	16,165
	SVM(*key substring-groups* + DF, tfidf-c)	$\mathbf{84.3}^{\text{accuracy}}$	28,726
Spanish (400)	No existing work has used this corpus yet.		
	SVM(*key substring-groups* + DF, tfidf-c)	$\mathbf{78.7}^{\text{accuracy}}$	2,519

As illustrated in Table 3, although the proposed algorithm (shown in gray background) **does** not use any preprocessing steps, such as *word segmentation* and stemming, it outperforms the character or word Ngrams based methods on three different language datasets. *Note that all these datasets are processed by the proposed algorithm in a **uniform** way rather than different **language-specific** ways.* Another observation is that the number of features (#Features) used in the proposed algorithm is larger than most other algorithms, which indicates that the promising performance of the proposed algorithm is at the cost of high feature dimension.

5.2 Multilingual Characteristics

Since the proposed algorithm treats the input documents as character sequences regardless of their syntax or semantic structures, no word segmentation technology is needed. Consequently, the proposed algorithm can deal with any language in any encoding, which has been demonstrated by the experiments in Table **3**.

Furthermore, the proposed algorithm is capable of handling text corpus containing both English and Chinese words at the same time. We conduct an experiment on the mixed-language dataset, including the "English_1400" corpus and 1,400 Chinese reviews (700 pos + 700 neg) randomly selected from the "Chinese_16000" corpus. Three-fold cross validation is adopted. The experimental results are shown in Figure 2. As is shown in Figure 2, the proposed algorithm achieves promising performance (shown in dark blue curve) on the mixed-language dataset, which is even better than the performance obtained by using only the English corpus.

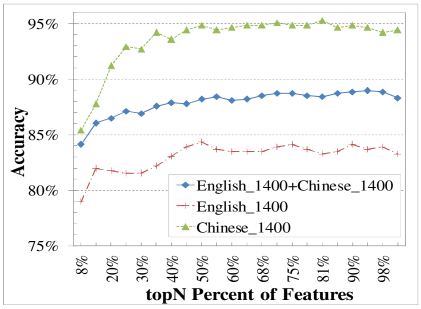


Fig. 2. The experiment on the mixed-language dataset (DF+"tfidf-c")

5.3 Feature Frequency vs. Feature Presence

The performance of sentiment classifier is highly affected by the text representation. To show the impact of the term weighting approaches, we conduct a series of experiments.

As demonstrated in Figure 3, different term weighting approaches lead to different classifying performances. Among all the term weighting methods, the "tfidf-c" outperforms all other approaches on the three open datasets in different languages, while the "tf" performs the worst. The "bool" always achieves better performance than the "three".

This observation agrees with Pang's finding: the better performance is achieved by accounting only for feature presence ("bool"), not feature frequency ("tf") [4]. However, the advanced feature frequency ("tfidf-c") is superior to the feature presence ("bool") in the proposed algorithm. Consequently, the "tfidf-c" is used in every experiment in the following subsections.

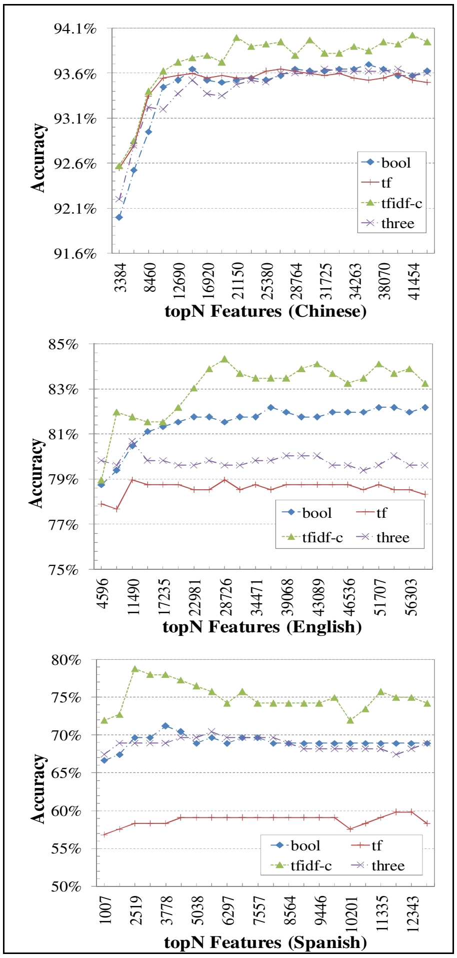
5.4 Influence of Feature Selecting

Figure **3** also displays the effectiveness of feature selecting to sentiment classification. As illustrated in Figure 3, the DF-based feature selection method can eliminate up to 50% or more of the unique substring-group features with either an improvement or no loss in classification accuracy, especially the "*tfidf-c*" curves (shown in green). In addition, Table 3 shows that all the best performances achieved by the proposed algorithm have used DF-based feature selection methods.

Based on above observations, we draw the following conclusions: the extracted substring-group features in step 1 are redundant and the feature selecting methods should be further used to eliminate the redundancy among the extracted substring-group features.

5.5 Transductive Learning vs. Inductive Learning

The following experiments show the effectiveness of using the transductive learning-based algorithm instead of inductive methods. Figure 4 gives the experimental results on the three open datasets in three different languages.



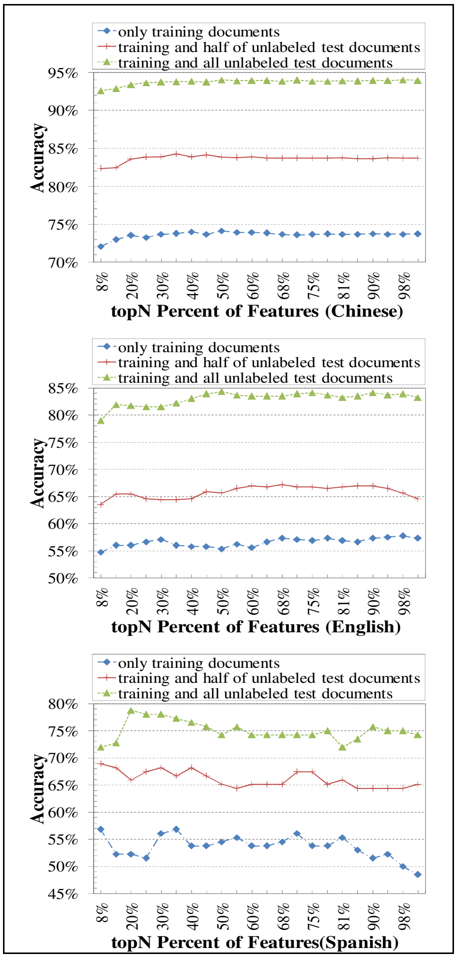


Fig. 3. The accuracies achieved by the proposed algorithm on three open sentiment datasets in different languages

Fig. 4. Comparisons of transductive learning (green and red) and inductive learning (dark blue)

As demonstrated in Figure 4, the transductive learning (shown in green and red curves) based algorithm is well situated for sentiment classification. With the growth of the unlabeled test documents added in the suffix tree construction, the performances of transductive learning based algorithms improve significantly.

Seen from the data shown in white background in Table 3 and the dark blue curves in Figure 4, another interesting observation is that the substring-group based algorithms in inductive learning setting is inferior to the algorithms using character or word Ngrams features, which illustrate transductive learning's importance to sentiment classification from another perspective.

The reason for the improvement by transductive learning is that the more unlabeled test documents are added to the construction of the suffix tree, the more complete the structure of suffix tree becomes. This, in turn, renders the suffix tree more representative of the text corpus. This leads to extracting more representative substring-group features from the suffix tree. Result is the converted documents being more representative of the original text documents. So the unlabeled test documents' structural information used at the beginning step indirectly contributes to the feature subsumption for sentiment classification.

6 Conclusion

In this study, both feature extracting and feature selecting are incorporated into sentiment classification, and the synergetic effect of them is studied. Moreover, the proposed algorithm combines the *substring-group* features with *transductive* learning.

Experiments have been conducted on three open datasets in *three* different languages, including *Chinese*, *English* and *Spanish*. The results show that the proposed algorithm achieves better performance than the existing algorithms, without any preprocessing steps (word segmentation, stemming, etc.). Furthermore, the proposed algorithm proves to be *multilingual*, and it can be directly used for sentiment classification with any language in any encoding. In terms of term weighting approaches, the "*tfidf-c*" performs best in the proposed algorithm. Experimental results also demonstrate that the *transductive* learning based algorithm can significantly improve the classifiers' performance by incorporating the *structural information of unlabeled test documents*.

In the future, we will examine the wrong classifications to get insights on how to improve the classifier. In addition, more feature extracting methods will be explored to improve the overall performance of sentiment classification.

Acknowledgments

This work is supported by National Natural Science Foundation of China (Grant No: 60405011, 60575057 and 60875073).

References

[1] Bo, P., Lillian, L.: Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval 2(1-2), 1–135 (2008)

[2] Bing, L.: Web data mining; Exploring hyperlinks, contents, and usage data. Springer, Heidelberg (2006)

[3] Bo, P., Lillian, L.: A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In: Proceedings of ACL (2004)

[4] Bo, P., Lillian, L., Shivakumar, V.: Thumbs up? Sentiment Classification using Machine Learning Techniques. In: Proceedings of EMNLP (2002)

[5] Ellen, R., Siddharth, P., Janyce, W.: Feature Subsumption for Opinion Analysis. In: Proceedings of EMNLP (2006)

[6] Tan, S., Zhang, J.: An empirical study of sentiment analysis for chinese documents. Expert Systems with Applications 34(4), 2622–2629 (2008)

[7] Raaijmakers, S., Kraaij, W.: A shallow approach to subjectivity classification. In: Proceedings of ICWSM (2008)

[8] Jun, L., Maosong, S.: Experimental Study on Sentiment Classification of Chinese Review using Machine Learning Techniques. In: Proceedings of IEEE NLPKE (2007)

[9] Dell, Z., Sun, L.W.: Extracting Key-Substring-Group Features for Text Classification. In: Proceedings of KDD, Philadelphia, PA (2006)
[10] Arnold, A., Nallapati, R., Cohen, W.: A comparative study of methods for transductive transfer learning. In: Proceedings of ICDM 2007 (2007)
[11] Xiaojin, Z.: Semi-supervised learning literature survey. Technical Report 1530, University of Wisconsin (2005)
[12] Sindhwani, V., Niyogi, P., Belkin, M.: Beyond the point cloud: from transductive to semi-supervised learning. In: Proceedings of ICML (2005)
[13] Joachims, T.: Transductive inference for text classification using support vector machines. In: Proceedings of ICML 1999 (1999)
[14] Vapnik, V.: Statistical Learning Theory. Wiley, NY (1998)
[15] Turney, P.D., Littman, M.L.: Unsupervised learning of semantic orientation from a hundred-billion-word corpus. Arxiv preprint cs.LG/0212012 (2002)
[16] Peter, T.: Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In: Proceedings of ACL (2002)
[17] Kim, S.-M., Eduard, H.: Determining the Sentiment of Opinions. In: Proceedings of COLING (2004)
[18] Minqing, H., Bing, L.: Mining Opinion Features in Customer Reviews. In: Proceedings of AAAI (2004)
[19] Xiaowen, D., Bing, L., Yu Philip, S.: A Holistic Lexicon-Based Approach to Opinion Mining. In: Proceedings of WSDM (2008)
[20] Alistair, K., Diana, I.: Sentiment Classification of Movie Reviews Using Contextual Valence Shifters. Computational Intelligence, Special Issue on Sentiment Analysis 22(2), 110–125 (2006)
[21] Ann, D., Khurshid, A.: Sentiment Analysis in Financial News: A Cohesion-based Approach. In: Proceedings of ACL (2007)
[22] Wan, X.: Using Bilingual Knowledge and Ensemble Techniques for Unsupervised Chinese Sentiment Analysis. In: Proceeding of EMNLP (2008)
[23] Kushal, D., Steve, L., David, P.: Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In: Proceedings of WWW (2003)
[24] Tony, M., Nigel, C.: Sentiment analysis using support vector machines with diverse information sources. In: Proceedings of EMNLP (2004)
[25] John, B., Mark, D., Fernando, P.: Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In: Proceedings of ACL (2007)
[26] Tan, S., Wang, Y., Cheng, X.: Combining learn-based and lexicon-based techniques for sentiment detection without using labeled examples. In: Proceedings of SIGIR (2008)
[27] Ukkonen, E.: On-line construction of suffix trees. Algorithmica 14(3), 249–260 (1995)
[28] Gusfield, D.: Algorithms on strings, trees, and sequences. Cambridge University Press, New York (1997)
[29] Thorsten, J.: A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In: Proceedings of ICML (1997)
[30] Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys (CSUR) 34(1), 1–47 (2002)
[31] Yang, Y., Pedersen, J.O.: A Comparative Study on Feature Selection in Text Categorization. In: Proceedings of ICML'97 (1997)
[32] Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features. Springer, Heidelberg (1997)

Decentralisation of ScoreFinder: A Framework for Credibility Management on User-Generated Contents

Yang Liao, Aaron Harwood, and Kotagiri Ramamohanarao

Computer Science and Software Engineering,
The University of Melbourne, Victoria, Australia
{liaoy,aaron,rao}@csse.unimelb.edu.au

Abstract. *User-generated content* (UGC) from Internet users has significant value only when its credibility can be established. A basic approach to establishing credibility is to take an average of scores from annotators, while more sophisticated approaches have been used to eliminate anomalous scoring behaviour by giving different weights to scores from different annotator profiles. A number of applications such as file sharing and article reviewing use a decentralised architecture. While computing a weighted average of static values in a decentralised application is well studied, sophisticated UGC algorithms are more complicated since source values to be aggregated and their weights may change in time. In our work we consider a centralised credibility management algorithm, ScoreFinder, as an example, and show both structured and unstructured approaches for computing time-dependent weighted average values in *peer-to-peer* (P2P) networks. Experimental results on two real data sets demonstrate that our approaches converge and deliver results comparable to those from the centralised version of ScoreFinder.

1 Introduction

User-Generated Content (UGC) is an increasingly important information source on the Web. UGC applications process individual data streams from a large number of Internet users and make this information available globally, e.g. Social Networking, Collaborative Content Publishing, File Sharing, Virtual Worlds and other collaborative activities. Examples of UGC include purported factual information, user opinions or reviews on public events and issues, files and documents. The value of the information from these applications is proportional to the information credibility – users need to be able to ascertain the credibility of the information in the UGC.

Confirming the credibility of a given content item using a centralised authority is infeasible due to the scale of UGC, and so most systems allow the users themselves to provide feedback, or score the content items that other users have provided. ScoreFinder [4] was proposed addressing the problem of aggregating feedback.

Not all UGC applications can be hosted centrally. For example, file sharing applications are more effective when they are decentralised, using a peer-to-peer (P2P) model, and it is widely accepted that the P2P model is applicable to a large range of UGC applications. Such applications are emerging, and they do not have a central trusted authority that will undertake the computations in a secure way. Indeed one of the motivations for these applications is to the contrary that there is no central authority in which trust must be placed. The distribution of trust therefore removes the so called "trust bottleneck" but creates a different problem of distributed trust. Furthermore, distributed functionality that benefits the majority of peers in a peer-to-peer system, such as UGC scoring, is likely to be supported in the sense that the required computational load put onto the peers will be accepted. As a result, sophisticated UGC credibility methods are very desirable but significantly challenging to apply in a decentralised system.

Both *structured* and *unstructured* peer-to-peer networks are considered in this paper, and different strategies are accordingly used. Structured networks provide unified access to each shared resource, namely all peers in a structured network can map an arbitrary resource identifier – usually file names or hash values of the file content – to a certain host. On the other hand, each peer in an unstructured peer-to-peer network sees only a local area of the network, and searches intended resources by propagating request messages.

In this paper we consider the problem of aggregating users' feedback for both a structured and an unstructured P2P architecture, and we use ScoreFinder as an instance of sophisticated credibility management algorithms. We provide a structured P2P approach for implementing ScoreFinder, and show how is it implemented on a structured peer-to-peer platform. We also provide an unstructured P2P approach for implementing ScoreFinder, based on gossiping. Particularly, the method of using time-slots defined on the Real-Time Clock to synchronise decentralised computation is novel and useful in other gossiping protocols where restricted synchronisation is required. We simulate network topology and churn to show its affect, as compared to a centralised or ideal computation.

2 Related Work

2.1 Article-Annotator Model and ScoreFinder

We in [4] introduce a model of credibility management, called the *Article-Annotator* (A-A) model and a comprehensive framework for Credibility Management, called *Score-Finder*. The participants and shared content items are named *Annotators* and *Articles* respectively, and the evaluations made by annotators are called *Scores*, which are numbers between 0 and 1.

The key operation of ScoreFinder, denoted as `ExpertnessEstimation`, is to iteratively calculate the weight of each score contributed by annotators to articles, and accordingly calculate the weighted average of scores to an article as the inference of the value or quality of the article, denoted as `AggregateResult`. An additional operation is to shift scores from each user to remove general biases, denoted as `ScoreShifting`.

2.2 Decentralised Frameworks

There are decentralised frameworks for implementing algorithms in structured and unstructured peer-to-peer networks. In unstructured p2p networks, there are algorithms employing flood or gossip messages to gradually aggregate values distributed over all peers. In [5], the authors proposed an approach to compute an overall average of values on each peer by continuously exchanging the status in each small-scaled sub-network (particularly, between the direct neighbourhoods). Similar approaches are adopted in GossipTrust [11], where opinions are uninterruptedly exchanged between each peer and a randomly selected peer (or all other peers) to reach the commonly agreed values. In [1], a general model of such problems is established for computing a linear combination of values distributed in peer-to-peer networks. More emphasis in the paper is placed on privacy of participants. On the other hand, most structured networks maintain a *Distributed Hash Table* (DHT) among peers. A DHT provides a many-to-one map between keys and peers; in most cases the distribution of keys over peers is uniformly random. It can be used for distributing data or for assigning logical roles over peers. Upon a given set of peers, a search to a given key always hits the same peer. These characteristics are employed in our framework, among other things, to map each of items to a peer that is in charge of organising the computation. Sophisticated frameworks also provide the ability of maintaining integrity of the Hash Table, by duplicating and/or migrating data items from left peers to other available peers. In practice, we used Pastry [9] as the DHT in our experiments; other DHT platforms like CAN [7] and Chord [10] provide the same function.

3 Decentralisation of ScoreFinder

3.1 Decentralisation in Structured Peer-to-Peer Networks

In this section we explain how we implemented a structured P2P application that uses ScoreFinder to rank articles. A decentralised architecture is proposed in Figure 1. The centralised ScoreFinder algorithm is simply split into two parts, the *annotator component* and the *tracker component* that could run on each peer; data exchange between such peers is through the underlying structured peer-to-peer network. The principles in splitting the centralised algorithm are: (1) to minimise the network transfer, and (2) to minimise computation on the tracker side because trackers are likely to become bottlenecks in the system. Thereby, the annotator component provides its expertness estimate and score shifting factor (respectively e and b) to each related tracker (i.e. trackers hosting articles that this annotator has read) and the tracker component provides the most recent results ($\mathbf{r} = \{r_j\}$) on hosted articles to the annotators that have read them.

Algorithm 1 and Algorithm 2 outline the computation in the tracker component and the annotator component, respectively. $\mathbf{S}^T$, $\mathbf{E}^T$ and $\mathbf{r}^T$ denote the relevant part (i.e. those for articles hosted in this peer) of $\mathbf{S}$, $\mathbf{E}$ and $\mathbf{r}$ in each tracker peer, while $\mathbf{S}^A$ and $\mathbf{r}^A$ denote the relevant part of $\mathbf{S}$ and $\mathbf{r}$ in each annotator peer. In particular, $\mathbf{r}^{AT}$ denotes the set of temporary results in the intersection of the corresponding $\mathbf{r}^A$ and $\mathbf{r}^T$. We use $\mathbf{B}^T$ to denote the score shifting vector (i.e. b_i) from annotators who gave scores to articles that are hosted by this tracker.

These two algorithms continue to iterate for the lifetime of the objects which instantiate them. We consider that in a peer-to-peer network, there are always new peers, new content items and new annotations to be included, so the process of refining the ranking

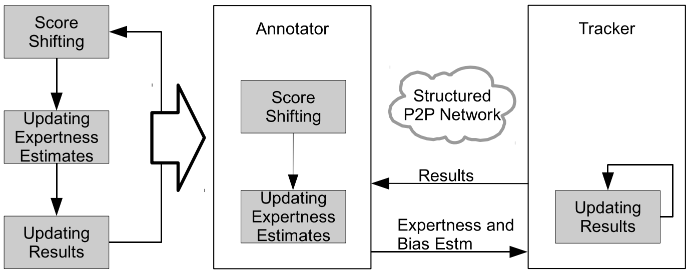


Fig. 1. Decentralising ScoreFinder on structured peer-to-peer networks

Algorithm 1. Process on Each Tracker

Require: All s_{ij} and score shifting factors
Ensure: Temporary result of each content item hosted by this tracker

```
1: loop
2:     $\mathbf{S^{T}}' \leftarrow$ ScoreShifting($\mathbf{S^T}, \mathbf{B^T}$)
3:     $\mathbf{r^T} \leftarrow$ AggregateResult($\mathbf{S^{T}}', \mathbf{E^T}$)
4:     for all connected annotators do
5:         SendToAnnotator($\mathbf{r^{AT}}$)
6:     end for
7:     Wait_A_Random_Period
8: end loop
```

Algorithm 2. Process on Each Annotator (i)

Require: All s_{ij} and the temporary result vector r
Ensure: The expertness and score shifting factor of this annotator

```
1: loop
2:     $e \leftarrow$ ExpertnessEstimation($\mathbf{r^A}, \mathbf{S^A}$)
3:     $b \leftarrow$ ScoreShiftingFactor($\mathbf{r^A}, \mathbf{S^A}$)
4:     for all connected trackers do
5:         SendToTracker($e, b$)
6:     end for
7:     Wait_A_Random_Period
8: end loop
```

is continuously running across the life cycle of the peer-to-peer application. Furthermore, there is no synchronisation between peers, i.e. every peer arbitrarily sends its most recent data at any time, and is always ready to receive messages from others. After receiving an update message, e.g. updating e or r_i, the peer updates its local cache, and uses the cache to continue with the next round of computation. Between each loop, the algorithm pauses for a short period. Network conditions largely influence the selection of this period. In our experiment, the period is set to 4 seconds since messages are delivered through an application level route, which usually consists of 2 or 3 hops.

3.2 Decentralisation in Unstructured Peer-to-Peer Networks

The ScoreFinder algorithm is also implemented in unstructured peer-to-peer networks using a gossip-based approach. We note that peers in unstructured peer-to-peer networks do not have a global perspective to the whole network, instead each peer sees a relatively small set of peers, called its *neighbours*. Therefore, there is no well-known peers, like trackers in a structured peer-to-peer network, which can be in charge of organising computation for each article; instead peers continuously exchange information with neighbours to propagate the influence of each original value to the whole network.

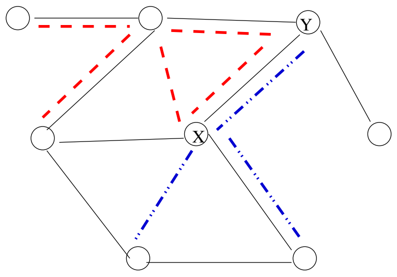
Also considered is the time-dependent change of values and their weights. Because of our expertise and score adjustment operations, the shifted scores ($\mathbf{S}^{\mathbf{T}'}$) and the expertise estimates for each annotator (e) change in time. We need to compute time-dependent weighted average values, as shown in the following formula, for coping such changes:

$$f(\mathbf{v}, \mathbf{w}, t) = \sum_i w_i(t) v_i(t), \tag{1}$$

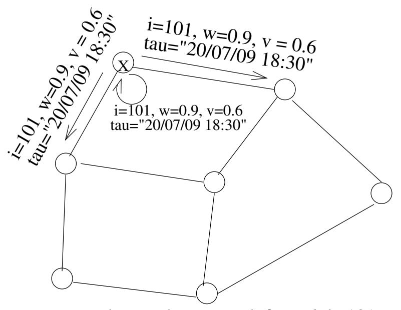
where $\mathbf{v} = \{v_i(t)\}$ and $\mathbf{w}(\mathbf{t}) = \{w_i(t)\}$ are two sets of time-depending functions. We define $\sum_i w_i(t) = 1$ for any t. A number of studies have focused on computing time-dependent weighted average values (as shown in Formula 1) in peer-to-peer networks using gossiped messages. The algorithm introduced in [3] uses epochs to divide the continuous computation into segments, in each segment the computation is restarted to validate changed values on each peer or values from new joiners. Peers in the network are synchronised by a broadcasting protocol.We then implemented ScoreFinder based on our time-slot based protocol.

Article Overlay Network. Considering the very large number of articles that are shared in current Internet applications, it is infeasible to store the status for all articles in each peer. In our approach, each node maintains a separate neighbour set for each annotated article, and exchanges messages only with peers that have the same annotated articles. This is equivalent to building an overlay network for each article over the original peer-to-peer network, and computing the weighted average score for the article on this overlay network. Figure 2(a) shows two overlay networks for article 101 and article 102 resp., built upon the original peer-to-peer network. Each contains all the peers giving scores to the article. Each peer finds peers that annotated same articles by flooding search messages.

Time-slots and Gossip Messages. To compute average values that always reflect the recent status of the peer-to-peer system, we use time-slots to synchronise the computation between peers. Time slots are periods that are predefined on the Real-Time Clock (RTC) and known by all peers. For example, the application could define every 5 minutes from 0:00 o'clock to be a time-slot, i.e. there are 288 slots in 24 hours. Nowadays, a large number of computers, including desktop computers, servers, mainframes and even mobile devices, could maintain a precise Real-Time Clock by frequently synchronising their local clock with Network Time Protocol (NTP) servers. Therefore, a time-slot is started and terminated nearly simultaneously on all such peers. In the beginning of each time-slot, computation is restarted from a new status, in which scores and weights are



(a) Overlay networks for articles are built upon the unstructured peer-to-peer network. It is shown that peer x and peer y have annotated both article 101 and article 102, hence they are in both overlay networks



The overlay network for article 101

(b) Gossip messages are sent between each pair of directly connected peers. The ID of the current time-slot is sent in each message along with i, w and v. It is shown that before sending the three messages, peer x has $v_{101} = 1.8$ and $w_{101} = 2.7$

Fig. 2. Unstructured network overlay

both updated based on the recent results from the last time-slot, and weighted average values reflecting those updated parameters are reached in the end of the time-slot.

Figure 2(b) shows an example of three gossip messages exchanged between peers. Each node maintains two values for each annotated articles (identified by i): the summed weights, w_i, and the summed product of weights and scores, v_i. Assuming that a node is directly connected to k peers for article i, in each step (the length of which is randomly determined by the node, usually 3-5 seconds) this node sends a message to each of the k peers as well as to itself. Each message consists of $i, \frac{v}{k+1}, \frac{w}{k+1}$ and the ID of the current time-slot (τ). Afterwards, this peer re-compute its w_i and v_i by adding v and w in all messages received in the past step that have the correct slot ID. It is noticeable that there could be zero or multiple messages from the same peer that are received in a step, which does not influence the validity of the algorithm. This cycle is repeated until the end of the current time-slot, then all peers update the result of each annotated article by computing $r_i^\tau = \frac{v_i}{w_i}$, and update their expertness (e^τ). Before starting the new time-slot, each peer re-evaluate its v_i and w_i by $(e^\tau s_{ij}^\tau)$ and e^τ. This process is shown in Algorithm 3.

4 Experiments

4.1 Outline of Experiment

In evaluating ScoreFinder, we implemented a simulator to imitate the scenario that content items are shared and annotated in peer-to-peer networks, including structured and unstructured models. In our experimental unstructured peer-to-peer network, the bootstrap is done by propagating messages in a flooding way to find peers that annotated same articles. Each peer keeps at least 5 neighbours for each overlay network (i.e. for

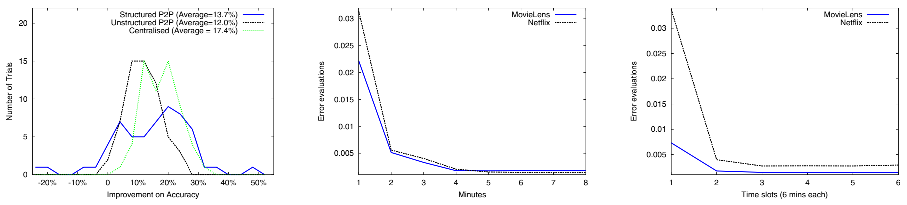
Algorithm 3. The Algorithm in each peer in a unstructured peer-to-peer network

```
$\tau \leftarrow$ NewTimeSlotID(); $e^{\tau} \leftarrow 0.5$; $\mathbf{S}^{\tau} \leftarrow \mathbf{S}$; $\mathbf{r}^{\tau} \leftarrow \mathbf{S}$
loop
    for all annotated articles : $i$ do
        $v_i \leftarrow e^{\tau} \times s_i^{\tau}$
        $w_i \leftarrow e^{\tau}$
    end for
    repeat
        for all annotated articles : $i$ do
            if there are peers connected for $i$ then
                for all connected peers and this peer do
                    SendMessage($i$, $\frac{v_i}{k+1}$, $\frac{w_i}{k+1}$, $\tau$)
                end for
                $v_i \leftarrow$sum of received $v$
                $w_i \leftarrow$sum of received $w$
            end if
        end for
        Clear_Received_Messages
        Wait_A_Random_Period
    until the end of the time-slot
    $\tau \leftarrow$ NewTimeSlotID()
    for all annotated articles : $i$ do
        if $w_i > 0$ then
            $r_i^{\tau} \leftarrow \frac{v_i}{w_i}$
        end if
    end for
    $\mathbf{S}^{\tau} \leftarrow$ ScoreShifting($\mathbf{S}, \mathbf{r}^{\tau}$)
    $e^{\tau} \leftarrow$ ExpertnessEstimation($\mathbf{S}^{\tau}, \mathbf{r}^{\tau}$)
end loop
```

each annotated article, a peer connects to 5 other peers giving scores to this article too), and propagates search messages through the p2p network in bootstrap or when a neighbour is found to be unavailable. Each search message has a Time-to-live (TTL) field to limit the diameter of the propagation. Each peer receiving the message retransmits the message to a limited number of neighbours until a peer annotating the same article is found or the TTL is exhausted. The targeted peer then sends back an acknowledgement message to the inquirer establishing a connection with it. We used the MovieLens data set [8], containing 10 million ratings for 10681 movies from 71567 volunteers,to evaluate our algorithms; the data set was shuffled and randomly re-sampled to a smaller data set in each trial. The two decentralised variants of ScoreFinder were examined in the experiment.

4.2 Evaluation Method

We evaluated our results with a supervised approach. In contrast of the size of our samples (150 annotators), we chose movies that received more than 2000 scores in each of the two data sets, so the average scores from such a large number of annotators could



(a) Distribution of improvement on accuracy to the baseline (on the MovieLens data set, in 60 trials)

(b) Convergence speed in structured peer-to-peer networks

(c) Convergence speed in unstructured peer-to-peer networks

Fig. 3. Performance evaluation on improvements to the baseline

be regarded as the common agreed opinion, and used as the reference. The evaluation function used in the experiment was the Mean Squared Error (MSE) between inferred credibility values and the reference; a smaller evaluation value denotes a more accurate result. In peer-to-peer networks, results on each peer could be different to each other, so we accumulated errors along each annotation, i.e. if an article is annotated by two peers, the error between the oracle score and the final result on each of the two peers is taken into the MSE.

4.3 Results and Discussion

Accuracy. Figure 3(a) shows the distribution of accuracy improvement in 60 trials, each was on a randomly re-sampled data set from respectively the MovieLens data set and the Netflix data set. There were 150 annotators and 2000 content items selected for building each re-sampled data set, as well as all scores between those selected entities. In this figure, our algorithms have a general better performance than the baseline. The performance of the unstructured peer-to-peer implementation stably follows the centralised implementation, where the performance of the structured peer-to-peer implementation has a larger variance to the centralised implementation. This could be explained by the difference on the extents of synchronisation of the two decentralised implementations. In the unstructured network, the algorithm closely follow the steps of the centralised ScoreFinder, namely each iteration is strictly synchronised by time-slots; whereas in the structured network, it does not follow the steps of the centralised ScoreFinder, namely all peers arbitrarily change their scores and weights any time, and the scores are gradually injected from the annotators to the trackers. This inconsistency made the latter one may have larger variance than the centralised algorithm. We also note that there is difference between average values calculated on different peers in the end of each time-slot, this led the accuracy of ScoreFinder in unstructured peer-to-peer networks consistently lower than the centralised implementation.

Looking into a single trial, the speed of convergence of the two ScoreFinder variants is analysed in Figure 3(b) and Figure 3(c). Both variants converged in the trial, nonetheless the unstructured variant consumed more time to reach convergence. This is because agreed weighted average values are reached after a round of exchanging messages in structured networks, but in unstructured networks it needs a whole time-slot to reach the agreed average values.

Robustness. To reveal the robustness of ScoreFinder in a variable network circumstance, we simulated two types of network conditions with different strengths. First, we randomly discarded messages between peers to examine the influence from packet loss. Second, we randomly shut down a number of peers in every 10 seconds to see how our algorithms reacted to variance on availability of peers. The same data set was used in the two simulations to facilitate comparison. Figure 4(a) and Figure 4(b) show the results in different packet loss rates and invalid peer ratios; in each condition the experiment was run 5 times. It is showed that the variant for unstructured networks is more sensitive to packet loss, whereas that for structured networks is more sensitive to churn.

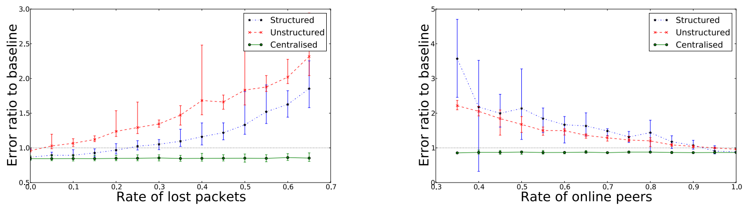


Fig. 4. Robustness testing; the error evaluations are showed by the relative ratios to the results from the baseline; the error-bars show the maximum and minimum relative errors in each condition

5 Conclusion

We introduced two decentralised variants of ScoreFinder, a credibility management framework, for respectively structured and unstructured peer-to-peer network applications. The performance of our algorithm is examined in an experiment using two real world data sets. The results reveal the merit of our approach by comparing to two baselines that are widely used in real applications. A number of issues regarding attacks and misbehaviour of users are discussed in the paper.

The primary challenge to decentralise ScoreFinder is to compute weighted average scores from parameters that change in time, by this means two approaches are used in different network models to synchronise distributed computation. In structured peer-to-peer networks, tracker components are hosted in peers which have addresses that are closest to the identification of articles, whereas globally agreed time-slots are used to coordinate paces of computation in unstructured peer-to-peer networks. The approach of using time-slots to synchronising computation across peers is equal to building a logic-time framework in a distributed environment, by which highly synchronised algorithms can be decentralised. Influence from churn and network conditions to the accuracy of the decentralisation methods was examined in the experiments.

5.1 Future Study

In this paper, no methods for identifying cliques and reducing their influence are considered. Annotators in cliques are only penalised by degrading their individual expertness levels considering that their scores may differ from other experts not in the clique. Nonetheless, unbiased scores from an annotator who gives biased scores to only a small subset of the content items are all lower weighted, hence we need an approach to say which part of their scores is unbiased and which part is not. Still, the criteria for clique identification could be closely related to the application, which is need to be further investigated.

In [6], the authors argue that weighted average values could be computed along spanning trees which is formed by multi-casting messages. We noticed that there are two advantages using spanning trees instead of exchanging gossip messages in implementing ScoreFinder. First, spanning tree-based algorithms have significantly faster convergence speed than gossip message approaches; furthermore, the upper bound of the number of messages to be exchanged before all peers reaching to an agreement is definite in a tree. Second, a peer in a spanning tree may change its local score and weight at any time, and the new agreement of weighted average value that reflects those changes will be reached in each peer under the same upper bound; namely no synchronising mechanisms, like time-slots or super peers, are necessary for spanning-tree styled algorithms. SCRIBE [2], CAN [7] and other multi-casting algorithms are useful platform, whereby spanning trees are built in unstructured peer-to-peer networks.

References

1. Bickson, D., Dolev, D., Bezman, G., Pinkas, B.: Peer-to-peer secure multi-party numerical computation. In: P2P '08: Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing, Washington, DC, USA, pp. 257–266. IEEE Computer Society Press, Los Alamitos (2008)
2. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.I.T.: SCRIBE: A large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in communications 20(8), 1489–1499 (2002)
3. Jelasity, M., Montresor, A., Babaoglu, O.: Gossip-based aggregation in large dynamic networks. ACM Trans. Comput. Syst. 23(3), 219–252 (2005)
4. Liao, Y., Harwood, A., Ramamohanarao, K.: Scorefinder: A method for topic sensitive credibility inference on documents. In: The 26th IEEE International Conference on Data Engineering, Long Beach, CA, USA (2010)
5. Mehyar, M., Spanos, D., Pongsajapan, J., Low, S.H., Murray, R.M.: Asynchronous distributed averaging on communication networks. IEEE/ACM Trans. Netw. 15(3), 512–520 (2007)
6. Ramabhadran, S., Ratnasamy, S., Hellerstein, J.M., Shenker, S.: Brief announcement: prefix hash tree. In: PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing, p. 368. ACM, New York (2004)
7. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A scalable content-addressable network. In: SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 161–172. ACM, New York (2001)

8. Riedl, J., Konstan, J.: Movielens data sets (July 2009), http://www.grouplens.org/node/73
9. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. LNCS, pp. 329–350. Springer, Heidelberg (2001)
10. Stoica, I., Morris, R., Karger, D., Frans Kaashoek, M., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 149–160. ACM, New York (2001)
11. Zhou, R., Hwang, K., Cai, M.: GossipTrust for Fast Reputation Aggregation in Peer-to-Peer Networks. IEEE Transactions on Knowledge and Data Engineering 20(9), 1282–1295 (2008)

Classification and Pattern Discovery of Mood in Weblogs

Thin Nguyen, Dinh Phung, Brett Adams, Truyen Tran, and Svetha Venkatesh

Curtin University of Technology
thin.nguyen@postgrad.curtin.edu.au,
{d.phung,b.adams,t.tran2,s.venkatesh}@curtin.edu.au

Abstract. Automatic data-driven analysis of mood from text is an emerging problem with many potential applications. Unlike generic text categorization, mood classification based on textual features is complicated by various factors, including its context- and user-sensitive nature. We present a comprehensive study of different feature selection schemes in machine learning for the problem of mood classification in weblogs. Notably, we introduce the novel use of a feature set based on the affective norms for English words (ANEW) lexicon studied in psychology. This feature set has the advantage of being computationally efficient while maintaining accuracy comparable to other state-of-the-art feature sets experimented with. In addition, we present results of data-driven clustering on a dataset of over 17 million blog posts with mood groundtruth. Our analysis reveals an interesting, and readily interpreted, structure to the linguistic expression of emotion, one that comprises valuable empirical evidence in support of existing psychological models of emotion, and in particular the dipoles pleasure–displeasure and activation–deactivation.

1 Introduction

Mood is a state of the mind such as being happy, sad or angry. It is a complex cognitive process which has received extensive research effort, and debate, among psychologists about its nature and structure [9,10,6]. But better scientific understanding of what constitutes a 'mood' has ramifications beyond psychology alone: for neuroscientists, it might offer insight into the functioning of the human brain; for medical professionals working in the domain of mental health, it might enable better monitoring and intervention for individuals and communities.

Research like that cited above aims to understand psychological drives and structures behind human mental states, and typically does so with expensive methodologies involving questionnaires or interviews that limit the number of participants. By contrast, our work aims to classify and cluster mood based on pre-existing content generated by users, which is collected unobtrusively – a sub-problem known as *mood analysis* in sentiment analysis [8]. Text-based mood classification and clustering, as a sub-problem of opinion and sentiment mining, have many potential applications identified in [8].

However, text-based mood analysis poses additional challenges beyond standard text categorization and clustering. The complex cognitive processes of mood formulation make it dependent on the specific social context of the user, their idiosyncratic associations of mood and vocabulary, syntax and style which reflect on language usage, or the specific genre of the text. In the case of weblogs, these challenges are highlighted by bloggers in the expression of diverse styles, relatively short text length, and informal language, such as jargon, abbreviations, and grammatical errors. This leads us to investigate whether machine learning-based feature selection methods for general text classification are still effective for blog text. Feature selection methods available in machine learning are often computationally expensive, relying on labeled data to learn discriminative features; but the blogosphere is vast (reaching almost 130 million[1]) and continuing to grow, making desirable a feature set that works without requiring supervised feature training to classify mood. To this end, we turn our attention to the result of a study that intersects psychology and linguistics known as affective norm for English words (ANEW) [1], and propose its use for mood classification.

In addition to classification, clustering mood into patterns is also an important task as it might provide vital clues about human emotion structure and has implications for sentiment-aware applications. While the structure of mood organization has been investigated from a psychological perspective for some time [9], to our knowledge, it has not been investigated from a data-driven and computational point of view. We provide an analysis of mood patterns using an unsupervised clustering approach and a dataset of more than 17 millions blog posts manually groundtruthed with users' moods.

Our contribution is twofold. First, we provide a comparative study of machine learning-based text feature selection for the specific problem of mood classification, elucidating insights into what can be transferred from a generic text categorization problem for mood classification. We then formulate a novel use of a psychology-inspired set of features for mood classification which does not require supervised feature learning, and is thus very useful for large-scale mood classification. Second, we provide empirical results for mood organization in the blogosphere on the largest dataset with mood groundtruth available today. To our knowledge, we are the first to consider the problem of data-driven mood pattern discovery at this scale.

The rest of this paper is organized as follows. Related work to feature selection methods for classification and clustering tasks in general text and in sentiment analysis is presented in Section 2. Work related to emotion measures in psychology is also examined in this section. Next, we present machine learning based feature selection schemes, together with the proposed ANEW feature set and linguistic analysis, applied to mood classification in two large datasets in Section 3. Section 4 presents the results of mood pattern discovery using an unsupervised learning technique, and is followed by some concluding remarks.

[1] From the state of the blogosphere 2008 at `http://technorati.com`

2 Related Background

For generic text categorization, a wide range of feature selection methods in machine learning has been studied. Most noticeably, Yang and Pedersen [14] conduct a comparative study on different feature selection schemes including information gain (IG), mutual information (MI), and χ^2 statistic (CHI).

Other than term-class interaction based, another approach for selecting features is to consider term statistics. Thresholds for term frequency (TF) or document frequency (DF) are commonly used in feature reduction in data mining. A joint from these two, the term frequency–inverse document frequency ($TF.IDF$) scheme, is also popular in text mining which often outperform TF and DF. Some works perform feature searching in narrower sets than over the entire vocabulary such as linguistic groups like the parts of speech (POS).

Related work making use of emotion bearing lexicon for sentiment analysis includes [2], where Dodds and Danforth use the valence values of ANEW [1] for estimating happiness levels in song lyrics, blogs, and the State of the Union.

Work related to mood clustering includes [5], where Leshed and Kaye group blog posts based on their moods to find mood synonymy.

Generally, emotions have been represented in dimensional and discrete perspectives. In the first methodology, emotion states are coded as combinations of some factors like valence and arousal. In contrast, the latter argues that each emotion has unique coincidence of experience, psychology, and behaviour [6]. We base our work on the dimensional mode for estimating the emotion sphere in blogosphere. Specifically, we use the circumplex model of affect [9,10] since it conceptualizes emotion states simply via valence and arousal dimensions, which can be computed using ANEW.

3 Textual-Based Mood Classification

3.1 Feature Selection Methods

Denote by $\mathcal{B}$ the corpus of all blogposts and by $\mathcal{M}=$ {sad, happy, ...} the set of all mood categories. In a standard feature selection setting, each blogpost $d \in \mathcal{B}$ is also labeled with a mood category $l_d \in \mathcal{M}$ and the objective is to extract from d a feature vector $\boldsymbol{x}^{(d)}$ being as discriminative as possible for d to be classified as l_d. For example, if we further denote by $\mathcal{V} = \left\{v_1, \ldots, v_{|V|}\right\}$ the set of all terms, then the feature vector $\boldsymbol{x}^{(d)} = [\ldots, \boldsymbol{x}_i^{(d)}, \ldots]$ might take a simple counting with its i-component $\boldsymbol{x}_i^{(d)}$ represents the number of times the term v_i appears in blogpost d, a scheme widely known as bag-of-word representation.

Term-based selection. These are features derived with respect to a term v. Two common features are term and document frequencies where *term frequency* $TF(v, d)$ represents the number of times the term v appears in document d, whereas *document frequency* $DF(v)$ is the number of blogposts containing the term v. It is also well-known in text mining that $TF.IDF\ (v, d)$ weighting scheme can potentially improve discriminative power where $TF.IDF(v,d) = TF(v,d) \times IDF(v)$ with $IDF(v) = |\mathcal{B}|/DF(v)$ is the inverse document frequency. In this work, a term v will be selected if it has high $DF(v)$ value, or high average values of $TF(v,d)$ or $TF.IDF(v,d)$ across all documents d over a threshold.

Term-Class interaction-based selection. The essence of these methods is to capture the dependence between terms and corresponding class labels during the feature selection process. Three common selection methods falling into this category are information gain $IG(v)$, mutual information $MI(v,l)$ and χ^2-statistics $CHI(v,l)$[14]. $IG(v)$ captures the information gain (measured in bits) when a term v is present or absent; $MI(v,l)$ measures the mutual information between a term v and a class label l; and lastly $CHI(v,l)$ measures the dependence between a term and a class label by comparing against one degree of freedom χ^2 distribution.

Affective Norms for English Words (*ANEW*). Apart from feature sets learned from data, for sentiment analysis, some emotion bearing lexicons have been subjectively chosen by labor power could help. Among them is ANEW [1], a set of 1034 sentiment conveying English words. These words are rated in terms of valence, arousal, and dominance they could convey. We apply the proposed set of 1034 words in ANEW exclusively as the feature vector, which means each blogpost is represented as a sparse counting vector for these ANEW words.

3.2 Mood Classification Results

It has been shown that linguistic components such as specific use of adverbs, adjectives or verbs can be a strong indicator for mood inference [8]. Therefore, in this paper, we further run a part-of-speech tagger to identify all terms that can be tagged as verbs, adjectives and adverbs. The tagger used is the SS-Tagger [12] ported to the Antelope NLP framework, giving a reasonable accuracy[2]. Three term weighting-based (*TF*, *DF*, *TF.IDF*) and three term-class interaction-based (*IG*, *MI*, *CHI*) selection methods are employed in this experiment. These feature selection methods shall be applied either on all terms (unigrams) or with respect to a subset of terms tagged with a specific *POS*.

Our experimental design is to compare and contrast which feature selection methods work best and to examine the effect of specific linguistic components in the context of mood classification. For classification methods, we have experimented with many off-the-shelf classifiers such as SVM, IBK, C4.5 and so on, however, the naive Bayes classifier (NBC) consistently outperforms these methods and therefore we shall only report the results w.r.t NBC. For each run, we use ten-fold cross-validation and repeat 10 runs and report the average result. To evaluate the results, we report two commonly-used measures: *accuracy* and *F-score* (which is measured based on *recall* and *precision*).

Effect of feature selection schemes and linguistic components. We use two datasets, namely WSM09 and IR05, for the task of mood classification. The first, WSM09, is provided by Spinn3r as the benchmark dataset for ICWSM

[2] www.proxem.com

2009 conference[3] which contains 44 millions blogposts crawled between August and October 2008. We extract a subset from this dataset consisting only blogposts from LiveJournal and query LiveJournal to obtain the mood groundtruth entered by the user when the posts were composed. We only consider the moods predefined by LiveJournal and discard others, resulting in approximately 600,000 blogposts. To validate the generalization of a feature selection scheme, we also run it on another dataset (IR05) created in [7] which contains 535,844 posts tagged the predefined moods. To make comparison with previous results in [11], we examine three popular moods {sad, happy, angry} in this experiment. The full set of 132 mood categories will be reported in the next section.

We run the experiment over combination of feature selection methods on different linguistic subsets and report the top ten best results in Table 1.

Table 1. Mood classification results for different feature selection schemes and for different part-of-speech subsets. Different combinations of feature selection methods and *POS* subsets are run, but we report only the top ten results sorted in ascending order of *F-score*.

WSM09				IR05			
Selection method	Linguistic subsets	Accuracy	F-score	Selection method	Linguistic subsets	Accuracy	F-score
	ANEW	0.713	**0.697**	IG	Adjective	0.738	0.709
IG	Verb	0.714	0.7		ANEW	0.734	**0.712**
TF.IDF	unigram	0.744	0.738	TF.IDF	AdjVbAdv	0.759	0.749
DF	AdjVbAdv	0.75	0.745	TF	AdjVbAdv	0.76	0.75
TF	AdjVbAdv	0.751	0.745	DF	AdjVbAdv	0.76	0.75
TF.IDF	AdjVbAdv	0.754	0.748	TF.IDF	unigram	0.765	0.756
DF	unigram	0.753	0.752	DF	unigram	0.765	0.762
TF	unigram	0.753	0.752	TF	unigram	0.765	0.762
IG	AdjVbAdv	0.762	0.756	IG	AdjVbAdv	0.773	0.763
IG	unigram	0.776	**0.774**	IG	unigram	0.791	**0.788**

With respect to feature selection scheme, information gain (*IG*) is observed to be the best selection scheme. Other term-class interaction based methods do not perform well, noticeably mutual information (*MI*) does not appear in any of the top ten results. These observations are consistent with what reported in [14] for text categorization problem. However, different with conclusions in [14], we found that *CHI* performs badly for mood classification task and does not appear in any top ten results. Surprisingly, both *TF* and *DF* performs better than *TF.IDF* in all-term (unigram) cases, which otherwise has been known oppositely in text mining that *IF.IDF* is often superior although much more computationally expensive. Thus, *TF* or *DF* should be the alternative candidates for *IG* for the trade-off of computational cost.

The performance of feature selection schemes experimented is also agreeable well across two datasets as can be seen in Table 1, except for the first few

[3] `http://www.icwsm.org/2009/data/`

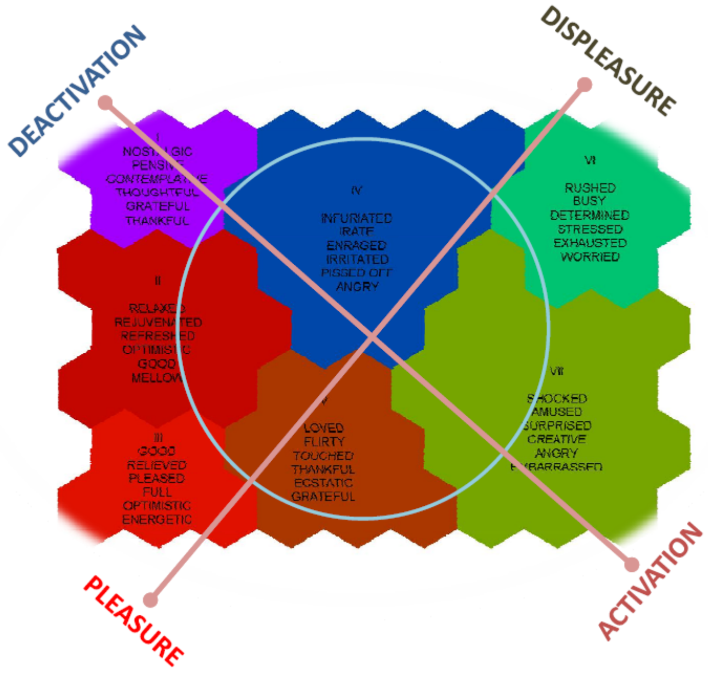


Fig. 1. Discovered mood structure map. Each cluster is annotated with the top six mood categories (best viewed in colour).

rows. Our best result stands at 77.4% *F-score* for WSM09 and 78.8% for IR05, which is higher than what reported in [11] (66.1%). With respect to the effect of linguistic components (which are not experimented in [11] and [5]), a combination of adjectives, verbs and adverbs (AdjVbAdv) dominates the top ten results and gives a very close performance to using all terms; noticeably using verbs or adjectives alone shows a good performance.

Performance of ANEW. Without the need of supervised feature selection stage, the result of ANEW feature is found to be very encouraging, appears in both top ten results across two datasets. The results across two datasets are also consistent, stand at approximately 70% *F-score* (still better than the best result reported in [11]).

4 Mood Pattern Discovery

While most of existing work has focused on supervised classification of mood, we are interested in discovering intrinsic patterns in mood structure using unsupervised learning approaches. Using a large, groundtruthed dataset of more than 17 millions posts introduced in [5], we aim to seek empirical evidences to answer various questions which have often posed in psychological studies. For example, does mood follow a continuum in its transition from 'pleasure' to 'displeasure', or from 'activation' to 'deactivation'? Is 'excited' closer to 'aroused' or 'happy'? Does 'depressed' transit to 'calm' before reaching 'happy'?

We use a total of 132 predefined moods defined by livejournal.com[4] for the clustering task. Given a corpus of more than 17 millions posts, it means that feature selection schemes presented in section **3.1** are very expensive to perform; for example, computing $MI(v, l)$ for each pair (term, mood label) will take $O\left(|\mathcal{M}| \times |\mathcal{V}|\right)$ where $|\mathcal{M}| = 132$ (number of moods) and $|\mathcal{V}|$ is the number of unique terms which could be in the order of hundreds of thousands. Since our results in section **3.2** have shown that the proposed ANEW feature set gives comparable results, marginally lower ($\sim$8%) in the classification compared to the best result but can totally avoid the expensive feature selection step, we shall employ ANEW as the feature vector in this section.

We choose multidimensional scaling, in particular, self-organizing map (SOM) [3] for clustering purpose. We use the SOM-PAK package [4] and the SOM Toolbox for Matlab [13] to train and visualize the map. For training, an 9×7 map is used which accounts for nearly a half of the mood classes. Using the recommendations in [3], the horizontal axis is roughly 1.3 that of the vertical axis; the node topology is hexagonal, and the number of training steps is 32,000 (about 500 times of the number of nodes).

Due to space restriction, we omit coarse-level results and present in the Figure 1 the structures of the clusters discovered in which top six moods in each cluster are included.

Several interesting patterns emerge from this analysis. At the highest level, one can observe the general transition of mood from an extreme of *pleasure* (clusters II, III, and V) to *displeasure* (clusters IV, VI, VII). On the *pleasure* polar we observe the moods having very high valence values[5] such as *good* (7.47), *loved* (8.64) or *relaxed* (7), whereas on the *displeasure* end, we observe the moods having low valence values such as *enraged* (2.46) or *stressed* (2.33). Certain mood transition is also evidential, for example the cluster path IV-II-III presents a transition pattern from *infuriated* to *relaxed* and then to *good*. Though not strongly emerging as in the case of *pleasure* $\leftrightarrow$ *displeasure,* a global pattern of *activation* $\leftrightarrow$ *deactivation* is also observed based on the analysis of the arousal measure as shown in Figure 1. Our results are indeed favorable of the core affect model for human emotion structure studied in psychology [9,10], generally agreeable with the global mood structure proposed in there.

5 Conclusion

We addressed the problem of mood classification and pattern discovery in weblogs. While the problem of machine learning based feature selection for text categorization has been intensively investigated, little work is found for textual based mood classification which is often more challenging. Our first contribution

[4] These moods can be viewed at `http://www.livejournal.com`

[5] Measured based on a study on ANEW reported in [1].

is a comprehensive comparison of different selection schemes across two large datasets. In addition, we propose a novel use of ANEW features which do not require a supervised selection phase, and thus, can be applied for mood analysis at a much larger scale. Our results have recalled similar findings in previous results, but also brought to light discoveries peculiar to the problem of mood classification. Our newly proposed feature set has also performed comparatively well at a fraction of the computational cost of supervised schemes, and was further validated by the results of an unsupervised clustering exercise, which clustered 17 million blog posts, and provided a unique view of mood patterns in the blogosphere. In particular, this study manifests global patterns of mood organization that are analogous to the pleasure–displeasure and activation–deactivation dimensions proposed independently in the psychology literature, such as the core affect model for the structure of human emotion. This data-driven organization of mood could be of interest to a wide range of practitioners in the humanities, and has many potential uses in sentiment-aware applications.

References

1. Bradley, M.M., Lang, P.J.: Affective norms for English words (ANEW): Stimuli, instruction manual and affective ratings. Technical report, The Center for Research in Psychophysiology, University of Florida (1999)
2. Dodds, P.S., Danforth, C.M.: Measuring the happiness of large-scale written expression: Songs, blogs, and presidents. Journal of Happiness Studies, 1–16 (2009)
3. Kohonen, T.: Self-Organizing Maps. Springer, Heidelberg (2001)
4. Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J.: SOM PAK: The self-organizing map program package. Technical report, Helsinki University of Technology (1996)
5. Leshed, G., Kaye, J.J.: Understanding how bloggers feel: recognizing affect in blog posts. In: Proc. of ACM Conf. on Human Factors in Computing Systems, CHI (2006)
6. Mauss, I.B., Robinson, M.D.: Measures of emotion: A review. Cognition & emotion 23(2), 209–237 (2009)
7. Mishne, G.: Experiments with mood classification in blog posts. In: Proc. of ACM Workshop on Stylistic Analysis of Text for Information Access, SIGIR (2005)
8. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval 2(1-2), 1–135 (2008)
9. Russell, J.A.: A circumplex model of affect. Journal of Personality and Social Psychology 39(6), 1161–1178 (1980)
10. Russell, J.A.: Emotion, core affect, and psychological construction. Cognition & Emotion 23(7), 1259–1283 (2009)
11. Sara, S., Lucy, V.: Sentisearch: Exploring mood on the web. In: Proc. of Workshop on Weblogs and Social Media, ICWSM (2009)
12. Tsuruoka, Y.: Bidirectional inference with the easiest-first strategy for tagging sequence data. In: Proc. of ACL Conf. on HLT/EMNLP, pp. 467–474 (2005)
13. Vesanto, J., Himberg, J., Alhoniemi, E., Parhankangas, J.: SOM toolbox for Matlab. Technical report, Helsinki University of Technology (2000)
14. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In: Proc. of ICML, pp. 412–420 (1997)

Capture of Evidence for Summarization: An Application of Enhanced Subjective Logic

Sukanya Manna, B. Sumudu U. Mendis, and Tom Gedeon

School of Computer Science, The Australian National University, Canberra, ACT 0200, Australia
{sukanya.manna,sumudu.mendis,tom.gedeon}@anu.edu.au

Abstract. In this paper, we present a method to generate an extractive summary from a single document using subjective logic. The idea behind our approach is to consider words and their co-occurrences between sentences in a document as evidence of their relatedness to the contextual meaning of the document. Our aim is to formulate a measure to find out 'opinion' about a proposition (which is a sentence in this case) using subjective logic in a closed environment (as in a document). Stronger opinion about a sentence represents its importance and are hence considered to summarize a document. Summaries generated by our method when evaluated with human generated summaries, show that they are more similar than baseline summaries.

Keywords: subjective logic, opinions, evidence, events, summarization, information extraction.

1 Introduction

It is sometime necessary to analyse a single document for intelligent decision making purpose in the absence of prior domain knowledge. In such a scenario, significant sentences from a document or rather gist of that document can only let an user know about what it is all about. Based on this filtered information, the user can decide what kind of measures to be taken to perform the analysis; thus, single-document summarization is one of the best ways to do this.

One way to do text summarization is by text extraction, which means to extract pieces of original text on statistical basis or heuristic methods and put them together to a new shorter text with as much information as possible preserved [9]. The concept of extracting significant sentences from a document for generating extractive summaries has drawn attention in the literature.

In this paper, our approach is not mere assigning scores to a sentence. When a document is looked from the perspective of human, they analyse it by finding what the main idea of the source text is and filtering what is essential in the information conveyed by the text. In [10], the authors have pointed out that a given piece of text is interpreted by different person in a different fashion especially in the way how they understand and interpret the context. Thus we see that human understanding and reasoning is *subjective* in nature unlike propositional logic which deals with either truth or falsity of a statement. So, to deal with this kind of situation we used *subjective logic* to find out sentences which are significant in the context and can be used to summarize a document.

2 Modeling 'Opinions' about a Sentence in a Document Using Subjective Logic

In this section, we present how we formulate 'opinion' about a sentence using subjective logic. Subjective logic [3] is a logic which operates on subjective beliefs about the world, and use the term opinion to denote the representation of a subjective belief. An opinion can be interpreted as a probability measure containing secondary uncertainty, and as such subjective logic can be seen as an extension of both probability calculus and binary logic. It is a type of probabilistic logic that explicitly takes uncertainty and belief into account. It is suitable for modeling and analysing situations involving uncertainty and incomplete knowledge [3], [4].

2.1 Interpretation of Evidence in a Document

How can we define evidence in a document? This is what we are building here automatically. We consider words, phrases or co-occurrence of words, or a sentence itself to be evidence present in a document. Now, based on this, our basic motivation is to formulate 'opinion' about a proposition, which is a sentence in this case. Stronger the opinions about a sentence, more is its significance in the document. These opinions are measured by probability expectation of a sentence. Greater the probability expectation, more significant is the sentence. If probability expectation of two sentences are similar, then we need to look at the sentence with lower uncertainty to fetch the important one [4] between two.

Assumptions: We propose the following framework for the practical application of subjective logic in a document computing context.
1. All the words or terms (removing the stop words) in a document are atomic.
2. The sentences are unique, i.e., each of them occur only once in a given document.

Representation of a document: A document consists of sentences. In this paper, a sentence is considered to be a set of words. In a document, sentences are separated by stop marks (".", "!", "?"). Terms (stop words excluded) are extracted and the frequencies (i.e. number of occurrences) of the words in each sentence are calculated.

Let us now define the notations which we will be using in the rest of the equations and explanations. Θ is the frame of discernment. We represent a document as a collection of words, which is

$$\Theta = D_w = \{w_1, w_2, ..., w_n\} \tag{1}$$

where, D_w is a document consisting of words like w_1, $w_2 ... w_n$ and $|D_w| = n$. Now,

$$\rho(\Theta) = \{\{w_1\}, \{w_2\}, ..., \{w_1, w_2, w_3, ..., w_n\}\} \equiv 2^{\Theta} \tag{2}$$

$$|\rho(\Theta)| = 2^n \tag{3}$$

Since a document is a collection of sentences, it can be represented as

$$D_s = \{s_1, s_2, ..., s_m\} \tag{4}$$

where m is a finite integer and each s_i is an element of $\rho(\Theta)$. Each sentence is comprised of words, which belong to the whole word collection of the document D_w. We thus represent each sentence by,

$$S_l = \{w_i w_k ... w_r\} \in \Theta \tag{5}$$

where, $1 \leq i,\ k,\ r \leq n$ and $S_l \in \rho(\Theta)$.

2.2 Definitions of 'Subjective Logic' and Our Conceptualization

In fig.1, we present a model of a document with four sentences (s_1, s_2, s_3, and s_4) and five words (w_1, w_2, w_3, w_4, and w_5) respectively. Let frequency of occurrence of each word in each sentence be one for simplicity. The words and sentences (atomic and non atomic states) represent evidence. Now, we use the original definitions from [4], and explain our formulation.

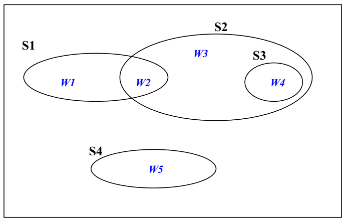


Fig. 1. Example of a document

The first step in applying evidential reasoning is to define a set of possible situations which is called the frame of discernment, Θ. A frame of discernment delimits a set of possible states of the world, exactly one of which is assumed to be true at any one time. In the given example, total number of all possible states are 2^5 for 5 words given.

Definition 1 (Belief Mass Assignment). *Let Θ be a frame of discernment. If with each substate $x \in 2^{\Theta}$ a number $m_{\Theta}(x)$ is associated such that:*

1. $m_{\Theta}(x) \geq 0$
2. $m_{\Theta}(\emptyset) = 0$
3. $\sum_{x \in 2^{\Theta}} m_{\Theta}(x) = 1$

then m_{Θ} is called a belief mass assignment in Θ, or BMA for short. For each substate $x \in 2^{\Theta}$, the number $m_{\Theta}(x)$ is called the belief mass of x.

We calculate BMA for each event by,

$$m(x) = \frac{F(x)}{Z}, \tag{6}$$

where $F(x) = \sum_{k=1}^{N} f_{x_k}$, where N is the total number of sentences in the document, $x \in 2^{\Theta}$, and f_{x_k} is the frequency of occurrence of event x in sentence k. In words, it is the total frequency of that event in all the sentences (or the whole document).

$$Z = \sum_{\substack{\forall x \neq \emptyset \\ f_x \neq 0}} F(x), \quad x \in 2^{\Theta} \tag{7}$$

Z is the total frequency of the all the events which has valid evidence of truth (whose frequency is non zero). In the given example, we have 7 valid states and their corresponding frequencies in the document are: $\{F(w_1) = 1,\ F(w_2) = 2,\ F(w_3) = 1,\ F(w_4) = 2,\ F(w_5) = 1, F(w_1, w_2) = 1,\ F(w_2, w_3, w_4) = 1\}$. Therefore, $Z = 9$ in this case. Using (6), we calculate BMA for each of the states (or events) in the given example.

Definition 2 (Belief Function). *Let Θ be a frame of discernment, and let m_Θ be a BMA on Θ. Then the belief function corresponding with m_Θ is the function $b :\ 2^{\Theta} \to [0, 1]$ defined by:*

$$b(x) = \sum_{y \subseteq x} m_\Theta(y), \quad x, y \in 2^{\Theta} \tag{8}$$

We calculate the belief of a sentence of the example as, $b(s_1) = m(w_1) + m(w_2) + m(w_1, w_2)$.

Definition 3 (Disbelief Function). *Let Θ be a frame of discernment, and let m_Θ be a BMA on Θ. Then the disbelief function corresponding with m_Θ is the function $d :\ 2^{\Theta} \to [0, 1]$ defined by:*

$$d(x) = \sum_{y \cap x = \emptyset} m_\Theta(y), \quad x, y \in 2^{\Theta}. \tag{9}$$

We calculate disbelief of s_1 by $d(s_1) = m(w_3) + m(w_4) + m(w_5)$.

Definition 4 (Uncertainty Function). *Let Θ be a frame of discernment, and let m_Θ be a BMA on Θ. Then the uncertainty function corresponding with m_Θ is the function $u :\ 2^{\Theta}\ [0, 1]$ defined by:*

$$u(x) = \sum_{\substack{y \cap x \neq \emptyset \\ y \nsubseteq x}} m_\Theta(y), \quad x, y \in 2^{\Theta}. \tag{10}$$

From Josang's idea, we can get the **Belief Function Additivity** which is expressed as:

$$b(x) + d(x) + u(x) = 1, \quad x \in 2^{\Theta},\ x \neq \emptyset. \tag{11}$$

Now, one can simply calculate the uncertainty of a sentence by using (11), i.e., $u(s_1) = 1 - (b(s_1) + d(s_1))$.

Definition 5 (Relative Atomicity). *Let Θ be a frame of discernment and let $x, y \in 2^{\Theta}$. Then for any given $y \neq \emptyset$ the relative atomicity of x to y is the function $a : 2^{\Theta} \rightarrow [0,1]$ defined by:*

$$a(x/y) = \frac{|x \cap y|}{|y|}, \quad x, y \in 2^{\Theta},\ y \neq \emptyset. \tag{12}$$

It can be observed that $x \cap y = \emptyset \Rightarrow a(x/y) = 0$ and that $y \subseteq x \Rightarrow a(x/y) = 1$. In all other cases relative atomicity will be a value between 0 and 1. The relative atomicity of an atomic state to its frame of discernment, denoted by $a(x/\Theta)$, can simply be written as $a(x)$. If nothing else is specified, the relative atomicity of a state then refers to the frame of discernment. In this case, we get the following relative atomicity for sentence s_1 as:

$a(s_1/w_1) = \frac{|s_1 \cap w_1|}{|w_1|} = \frac{1}{1} = 1$
$a(s_1/w_2) = \frac{|s_1 \cap w_2|}{|w_2|} = \frac{1}{1} = 1$
$a(s_1/\{w_1, w_2\}) = a(s_1, s_1) = \frac{|s_1 \cap \{w_1, w_2\}|}{|\{w_1, w_2\}|} = \frac{2}{2} = 1...$
$a(s_1/w_5) = a(s_1/s_4) = \frac{|s_1 \cap w_5|}{|w_5|} = \frac{0}{1} = 0$
Likewise, we calculate the atomicity for other sentences.

Definition 6 (Probability Expectation). *Let Θ be a frame of discernment with BMA m_{Θ} then the probability expectation function corresponding with m_{Θ} is the function $E : 2^{\Theta} \rightarrow [0,1]$ defined by:*

$$E(x) = \sum_{y} m_{\Theta}(y) a(x/y), \quad y \in 2^{\Theta}. \tag{13}$$

So, for the given example, we calculate *ProbExp* for sentence s_1 as follows:
$E(s_1) = m(w_1)a(s_1/w_1) + m(w_2)a(s_1/w_2) + m(\{w_1, w_2\})a(s_1/\{w_1, w_2\}) + ... + m(w_5)a(s_1/w_5)$ For compactness and simplicity of notation we will in the following denote belief, disbelief, uncertainty, relative atomicity and opinion functions as b_x, d_x, u_x, a_x and ω_x respectively. Thus opinion (ω_{s_1} or $\omega(s_1)$)about a sentence s_1 can be expressed using these four parameters as, $\omega(s_1) = (b(s_1),\ d(s_1),\ u(s_1),\ a(x))$.

In this context, we order sentences based on descending order of their probability expectation and ascending order of their uncertainty; sentence with stronger 'opinion' has greater significance in a document.

3 Method

3.1 Data Processing

In this experiment we used DUC2001 data set [1] for evaluation. The documents are grouped based on a specific topic. Our main aim is to see how our model works on single documents for content analysis purposes, so we focussed on this kind of data set unlike other information retrieval areas. These documents were parsed, tokenized, cleaned, and stemmed. The cleaning is done by removing the stop words. DUC2001 comes with human generated summaries and baseline summaries, providing a good platform for evaluation.

3.2 Generation of Summaries

Summaries are broadly classified into text extraction and text abstraction [7], [5]. For text extraction, sentences from the documents are used as summaries and for text abstraction important pieces of information are extracted and then stitched together to form summaries following some linguistic rules. This evidence based model can be used as a text extraction as we use the original sentences. We compared our method and DUC baseline summaries with the human generated summaries provided by them.

Evidence based model (PEU): In sec.2, we described our method of sentence ranking; subjective logic based where we ranked the sentences based on the descending order probability expectation and ascending order of uncertainty (PEU) of that sentence in the document. We took 30% [2] of the top ranked sentences and used them as summary.

3.3 Evaluation by ROUGE

ROUGE [6] stands for Recall-Oriented Understudy for Gisting Evaluation. It includes measures to automatically determine the quality of a summary by comparing it to other (ideal) summaries created by humans. ROUGE is a recall based metric for fixed length summaries. The measures count the number of over lapping units such as n-gram, word sequences, and word pairs between the computer-generated summary to be evaluated and the ideal summaries created by humans.

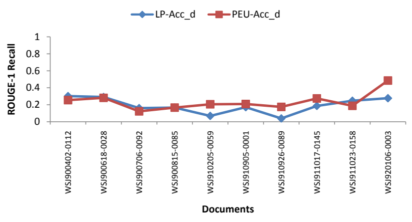
In this experiment, we present the result with ROUGE-1 (n-gram, where n=1) at 95% confidence level. ROUGE is sensitive to the length of the summaries [8]; hence we fixed the length to 100 words for the evaluation.

4 Results

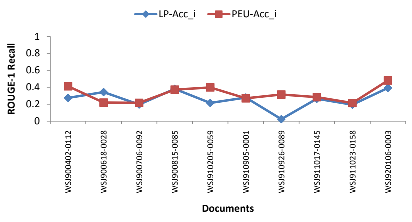
We used DUC2001 dataset for this experiment. Among different document sets, we presented here the evaluation with ‘daycare’, ‘healthcare’ and ‘pres92’. We compared our method (PEU) and baseline summaries (denoted by LP) with two different human assessors. For each set the assessors are different. The average (table.1) results show that our method out performs the baseline summaries.

Figures 2(a) and 2(b) show the evaluation comparison using daycare data, where Acc_d and Acc_i are two human assessors. In both the figures 2(a) and 2(b), our method outperforms the baseline summaries (90% of documents with Acc_d and 80% of documents with Acc_i).

Now, figures 3(a) and 3(b) present the results with healthcare data set. Here Acc_b and Acc_j are the two human assessors. There was no baseline summary for the 7th document in this series. So, in both the figures we have value as 0 in the comparison results. In fig.3(a), ROUGE score for PEU with Acc_b is higher

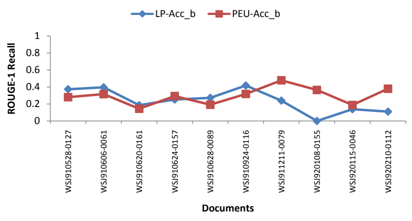


(a) Comparison with Assessor d

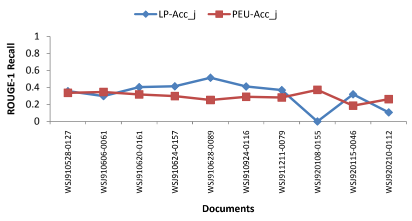


(b) Comparison with Assessor i

Fig. 2. Daycare dataset

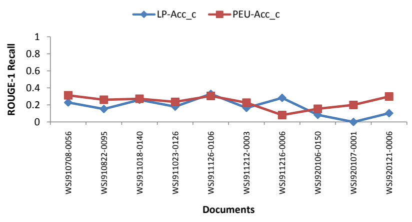


(a) Comparison with Assessor b

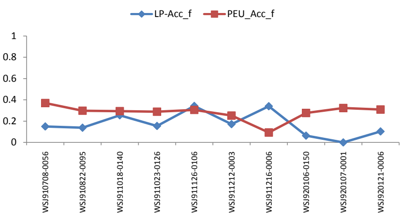


(b) Comparison with Assessor j

Fig. 3. Healthcare dataset



(a) Comparison with Assessor c



(b) Comparison with Assessor f

Fig. 4. Pres92 dataset

than baseline on average except for 30% of the documents. In fig.3(b), baseline shows higher similarity with Acc_j than PEU in 60% documents.

In figures 4(a) and 4(b), PEU outperforms the baseline summaries (90% of documents with each assessors). In table.1, Acc1 and Acc2 are alias of human assessors used in each case. Except for Acc2 in healthcare data, PEU has outperformed all the baseline summaries.

From these results, we can see that summaries produced by humans are abstract. So overlap with human generated of summaries with automated ones can vary a lot unless they are compared with extractive summaries created by humans selecting the original sentences from documents.

Table 1. Summary of all three sets of results (ROUGE-1 Recall)

	LP-Acc1	LP-Acc2	PEU-Acc1	PEU-Acc2
daycare	0.19	0.26	0.24	0.32
healthcare	0.27	0.35	0.29	0.29
*pres*92	0.20	0.19	0.24	0.28
Avg	0.22	0.27	0.25	0.29

5 Conclusion

In this paper we presented an evidence based sentence extraction method for single document summarization. We used enhanced subjective logic to formulate the whole process. Here standard methods for evaluating data are used; in the whole process we figured out that summarization is subjective to the user. In our system we basically used word frequency and co-occurrence concept for formulating subjective logic; rather superficial knowledge. But the results are good in the sense that they have outperformed baseline summaries as illustrated in the results. For our future work we will extend this method to perform deeper semantic analysis of the text and redefine some features of subjective logic in document computing context.

References

1. DUC (2001), The document understanding conference, http://duc.nist.gov/
2. Dalianis, H.: SweSum-A Text Summarizer for Swedish (2000), http://www.dsv.su.se/%7Ehercules/papers.Textsumsummary.html
3. Jøsang, A.: Artificial reasoning with subjective logic. In: Proceedings of the Second Australian Workshop on Commonsense Reasoning, vol. 48 (1997) Perth:[sn]
4. Jøsang, A.: A logic for uncertain probabilities. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 9(3), 279–311 (2001)
5. Liddy, E.D.: The discourse-level structure of empirical abstracts: an exploratory study. Inf. Process. Manage. 27(1), 55–81 (1991)
6. Lin, C.-Y.: Rouge: A package for automatic evaluation of summaries, Barcelona, Spain, July 2004, pp. 74–81. Association for Computational Linguistics (2004)
7. Lin, C.Y., Hovy, E.: Identifying topics by position. In: Proceedings of the fifth conference on Applied natural language processing, pp. 283–290. Morgan Kaufmann Publishers Inc., San Francisco (1997)
8. Lin, C.Y., Hovy, E.: Automatic evaluation of summaries using n-gram co-occurrence statistics. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, vol. 1, p. 78. Association for Computational Linguistics (2003)
9. Mani, I., Maybury, M.T.: Advances in automatic text summarization. MIT Press, Cambridge (1999)
10. Pardo, T.A.S., Rino, L.H.M., Nunes, M.G.V.: Extractive summarization: how to identify the gist of a text. In: The Proceedings of the 1st International Information Technology Symposium–I2TS, Citeseer, pp. 1–6 (2002)

Fast Perceptron Decision Tree Learning from Evolving Data Streams

Albert Bifet, Geoff Holmes, Bernhard Pfahringer, and Eibe Frank

University of Waikato, Hamilton, New Zealand
{abifet,geoff,bernhard,eibe}@cs.waikato.ac.nz

Abstract. Mining of data streams must balance three evaluation dimensions: accuracy, time and memory. Excellent accuracy on data streams has been obtained with Naive Bayes Hoeffding Trees—Hoeffding Trees with naive Bayes models at the leaf nodes—albeit with increased runtime compared to standard Hoeffding Trees. In this paper, we show that runtime can be reduced by replacing naive Bayes with perceptron classifiers, while maintaining highly competitive accuracy. We also show that accuracy can be increased even further by combining majority vote, naive Bayes, and perceptrons. We evaluate four perceptron-based learning strategies and compare them against appropriate baselines: simple perceptrons, Perceptron Hoeffding Trees, hybrid Naive Bayes Perceptron Trees, and bagged versions thereof. We implement a perceptron that uses the sigmoid activation function instead of the threshold activation function and optimizes the squared error, with one perceptron per class value. We test our methods by performing an evaluation study on synthetic and real-world datasets comprising up to ten million examples.

1 Introduction

In the data stream model, data arrive at high speed, and algorithms that process them must do so under very strict constraints of space and time. Consequently, data streams pose several challenges for data mining algorithm design. First, algorithms must make use of limited resources (time and memory). Second, they must deal with data whose nature or distribution changes over time.

An important issue in data stream mining is the cost of performing the learning and prediction process. As an example, it is possible to buy time and space usage from cloud computing providers [25]. Several rental cost options exist:

– Cost per hour of usage: Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. Cost depends on the time and on the machine rented (small instance with 1.7 GB, large with 7.5 GB or extra large with 15 GB).
– Cost per hour and memory used: GoGrid is a web service similar to Amazon EC2, but it charges by RAM-Hours. Every GB of RAM deployed for 1 hour equals one RAM-Hour.

It is crucial to find mining methods that use resources efficiently. In this spirit, we propose in this paper the *Hoeffding Perceptron Tree* for classification, as a faster method compared to the state-of-the-art Hoeffding Tree with naive Bayes leaves. The idea is to implement perceptron classifiers at the leaves of the Hoeffding Tree, to potentially increase accuracy, but mainly to reduce runtime.

We introduce the use of RAM-Hours as an evaluation measure of the resources used by streaming algorithms. The paper is structured as follows: related work is presented in Section 2. Hoeffding Perceptron Trees and bagging of such trees are discussed in Section 3. An experimental evaluation is conducted in Section 4. Finally, conclusions and suggested items for future work are presented in Section 5.

2 Related Work

Standard decision tree learners such as ID3, C4.5, and CART [18,21] assume that all training examples can be stored simultaneously in main memory, and are thus severely limited in the number of examples they can learn from. In particular, they are not applicable to data streams, where potentially there is no bound on the number of examples and these arrive sequentially.

Domingos et al. [6,14] proposed the *Hoeffding tree* as an incremental, anytime decision tree induction algorithm that is capable of learning from data streams, assuming that the distribution generating examples does not change over time.

Hoeffding trees exploit the fact that a small sample can often suffice to choose a splitting attribute. This idea is supported by the Hoeffding bound, which quantifies the number of observations (in our case, examples) needed to estimate some statistics within a prescribed precision (in our case, the goodness of an attribute). More precisely, the Hoeffding bound states that with probability $1-\delta$, the true mean of a random variable of range R will not differ from the estimated mean after n independent observations by more than:

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}.$$

A theoretically appealing feature of Hoeffding Trees not shared by other incremental decision tree learners (ID4 [22], ID5 [24]) is that it has sound guarantees of performance. Using the Hoeffding bound one can show that its output is asymptotically nearly identical to that of a non-incremental learner using infinitely many examples. CVFDT [14] is an extension of the Hoeffding Tree to evolving data streams, but does not exhibit theoretical guarantees.

Outside the data stream world, there is prior work on using perceptrons or similar classifiers in decision trees. Utgoff [24] presented the *Perceptron Decision Tree* as a decision tree in which each leaf node uses the perceptron as a classifier of the input instances. Bennett et al. [2] showed that maximizing margins in perceptron decision trees can be useful to combat overfitting. Zhou [26] proposed *Hybrid Decision Trees* as a hybrid learning approach combining decision trees with neural networks.

Frank et al. [7] investigated using *Model Trees* for classification problems. Model trees are decision trees with linear regression functions at the leaves.

Logistic Model Trees [17] are model trees that use logistic regression instead of linear regression. They have been shown to be very accurate and compact classifiers, but their induction is very time-consuming.

The LTree algorithm of Gama [8] embodies a general framework for learning functional trees, multivariate classification or regression trees that can use combinations of attributes at decision nodes, leaf nodes, or both.

In the data streams literature, Ikonomovska et al. [15] presented FIMT, a fast incremental model tree for regression on static data streams. To deal with concept drift, Ikonomovska et al. [16] proposed FIRT-DD as an adaption of the FIMT algorithm to time-changing distributions. FIMT and FIRT-DD use a perceptron learner at the leaves to perform regression. Considering classification methods for data streams, Bifet et al. [4] presented two new ensemble learning methods: one using bagging with decision trees of different size and one using `ADWIN`, an adaptive sliding window method that detects change and adjusts the size of the window correspondingly. We revisit the latter approach in this paper.

3 Perceptrons and Hoeffding Perceptron Trees

In this section, we present the perceptron learner we use, and the Hoeffding Perceptron Tree based on it. We also consider bagging trees with change detection.

3.1 Perceptron Learning

We use an online version of the perceptron that employs the sigmoid activation function instead of the threshold activation function and optimizes the squared error, with one perceptron per class value.

Given a data stream $\langle \boldsymbol{x}_i, y_i \rangle$, where $\boldsymbol{x}_i$ is an example and y_i is its example class, the classifier's goal is to minimize the number of misclassified examples. Let $h_{\boldsymbol{w}}(\boldsymbol{x}_i)$ be the hypothesis function of the perceptron for instance $\boldsymbol{x}_i$. We use the mean-square error $J(\boldsymbol{w}) = \frac{1}{2}\sum(y_i - h_{\boldsymbol{w}}(\boldsymbol{x}_i))^2$ instead of the 0-1 loss function, since it is differentiable.

The classic perceptron takes a linear combination and thresholds it. The predicted class of the perceptron is $h_{\boldsymbol{w}}(\boldsymbol{x}_i) = \text{sgn}(\boldsymbol{w}^T\boldsymbol{x}_i)$, where a bias weight with constant input is included. Our hypothesis function $h_{\boldsymbol{w}} = \sigma(\boldsymbol{w}^T\boldsymbol{x})$ instead uses the sigmoid function $\sigma(x) = 1/(1+e^{-x})$ since it has the property $\sigma'(x) = \sigma(x)(1-\sigma(x))$. Thus, we can compute the gradient of the error function

$$\nabla J = -\sum_i (y_i - h_{\boldsymbol{w}}(\boldsymbol{x}_i))\nabla h_{\boldsymbol{w}}(\boldsymbol{x}_i)$$

where for sigmoid hypothesis

$$\nabla h_{\boldsymbol{w}}(\boldsymbol{x}_i) = h_{\boldsymbol{w}}(\boldsymbol{x}_i)(1 - h_{\boldsymbol{w}}(\boldsymbol{x}_i))$$

obtaining the following weight update rule

$$\boldsymbol{w} = \boldsymbol{w} + \eta\sum_i (y_i - h_{\boldsymbol{w}}(\boldsymbol{x}_i))h_{\boldsymbol{w}}(\boldsymbol{x}_i)(1 - h_{\boldsymbol{w}}(\boldsymbol{x}_i))\boldsymbol{x}_i$$

Because we work in a data stream scenario, rather than performing batch updates, we use stochastic gradient descent where the weight vector is updated after every example. As we deal with multi-class problems, we train one perceptron for each class. To classify an unseen instance $\boldsymbol{x}$, we obtain the predictions $h_{\boldsymbol{w_1}}(\boldsymbol{x}), \ldots, h_{\boldsymbol{w_n}}(\boldsymbol{x})$ from the perceptrons, and the predicted class is $\arg\max_{class} h_{\boldsymbol{w_{class}}}(\boldsymbol{x})$. The pseudocode is shown in Figure 1.

```
Perceptron Learning($Stream, \eta$)
1  for each class
2        do Perceptron Learning($Stream, class, \eta$)

Perceptron Learning($Stream, class, \eta$)
1  ▷ Let $w_0$ and $\boldsymbol{w}$ be randomly initialized
2  for each example $(\boldsymbol{x}, y)$ in Stream
3        do if $class = y$
4              then $\delta = (1 - h_{\boldsymbol{w}}(\boldsymbol{x})) \cdot h_{\boldsymbol{w}}(\boldsymbol{x}) \cdot (1 - h_{\boldsymbol{w}}(\boldsymbol{x}))$
5              else $\delta = (0 - h_{\boldsymbol{w}}(\boldsymbol{x})) \cdot h_{\boldsymbol{w}}(\boldsymbol{x}) \cdot (1 - h_{\boldsymbol{w}}(\boldsymbol{x}))$
6           $\boldsymbol{w} = \boldsymbol{w} + \eta \cdot \delta \cdot \boldsymbol{x}$

Perceptron Prediction($\boldsymbol{x}$)
1  return $\arg\max_{class} h_{\boldsymbol{w}_{class}}(\boldsymbol{x})$
```

Fig. 1. Perceptron algorithm

3.2 Hoeffding Perceptron Tree

Hoeffding trees [6] are state-of-the-art in classification for data streams and they perform prediction by choosing the majority class at each leaf. Their predictive accuracy can be increased by adding naive Bayes models at the leaves of the trees. However, Holmes et al. [12] identified situations where the naive Bayes method outperforms the standard Hoeffding tree initially but is eventually overtaken. They propose a hybrid adaptive method that generally outperforms the two original prediction methods for both simple and complex concepts. We call this method *Hoeffding Naive Bayes Tree* (**hnbt**). This method works by performing a naive Bayes prediction per training instance, and comparing its prediction with the majority class. Counts are stored to measure how many times the naive Bayes prediction gets the true class correct as compared to the majority class. When performing a prediction on a test instance, the leaf will only return a naive Bayes prediction if it has been more accurate overall than the majority class, otherwise it resorts to a majority class prediction.

We adapt this methodology to deal with perceptrons rather than naive Bayes models. A *Hoeffding Perceptron Tree* (**hpt**) is a Hoeffding Tree that has a perceptron at each leaf. Similarly to **hnbt**, predictions by the perceptron are only used if they are more accurate on average than the majority class. It improves on **hnbt** in terms of runtime because it does not need to estimate the statistical distribution for numeric attributes and calculate density values based on the exponential function, and for discrete attributes it does not need to calculate divisions to estimate probabilities.

Finally, a *Hoeffding Naive Bayes Perceptron Tree* (**hnbpt**) is a Hoeffding Tree that has three classifiers at each leaf: a majority class, naive Bayes, and a perceptron. Voting is used for prediction. It is slower than the Hoeffding Perceptron Tree and the Hoeffding Naive Bayes Tree, but it combines the predictive power of the base learners.

3.3 Bagging Trees with ADWIN

ADWIN [3] is a change detector and estimator that solves in a well-specified way the problem of tracking the average of a stream of bits or real-valued numbers. ADWIN keeps a variable-length window of recently seen items, with the property that the window has the maximal length statistically consistent with the hypothesis "there has been no change in the average value inside the window".

ADWIN is parameter- and assumption-free in the sense that it automatically detects and adapts to the current rate of change. Its only parameter is a confidence bound δ, indicating how confident we want to be in the algorithm's output, inherent to all algorithms dealing with random processes.

Also important for our purposes, ADWIN does not maintain the window explicitly, but compresses it using a variant of the exponential histogram technique. This means that it keeps a window of length W using only $O(\log W)$ memory and $O(\log W)$ processing time per item.

ADWIN Bagging is the online bagging method of Oza and Russell [19] with the addition of the ADWIN algorithm as a change detector. When a change is detected, the worst classifier of the ensemble of classifiers is removed and a new classifier is added to the ensemble.

4 Comparative Experimental Evaluation

Massive **O**nline **A**nalysis (MOA) [13] is a software environment for implementing algorithms and running experiments for online learning from data streams. All algorithms evaluated in this paper were implemented in the Java programming language by extending the MOA software.

We use the experimental framework for concept drift presented in [4]. Considering data streams as data generated from pure distributions, we can model a concept drift event as a weighted combination of two pure distributions that characterizes the target concepts before and after the drift. This framework defines the probability that a new instance of the stream belongs to the new concept after the drift based on the sigmoid function.

Definition 1. *Given two data streams a, b, we define $c = a \oplus_{t_0}^{W} b$ as the data stream built by joining the two data streams a and b, where t_0 is the point of change, W is the length of change,* $\Pr[c(t) = b(t)] = 1/(1 + \mathrm{e}^{-4(t-t_0)/W})$ *and* $\Pr[c(t) = a(t)] = 1 - \Pr[c(t) = b(t)]$.

In order to create a data stream with multiple concept changes, we can build new data streams joining different concept drifts, i. e. $(((a \oplus_{t_0}^{W_0} b) \oplus_{t_1}^{W_1} c) \oplus_{t_2}^{W_2} d) \ldots$

4.1 Datasets for Concept Drift

Synthetic data has several advantages – it is easier to reproduce and there is little cost in terms of storage and transmission. For this paper we use the data generators most commonly found in the literature.

SEA Concepts Generator. This artificial dataset contains abrupt concept drift, first introduced in [23]. It is generated using three attributes, where only the two first attributes are relevant. All the attributes have values between 0 and 10. The points of the dataset are divided into 4 blocks with different concepts. In each block, the classification is done using $f_1 + f_2 \leq \theta$, where f_1 and f_2 represent the first two attributes and θ is a threshold value. The most frequent values are 9, 8, 7 and 9.5 for the data blocks. In our framework, SEA concepts are defined as follows:

$$(((SEA_9 \oplus^W_{t_0} SEA_8) \oplus^W_{2t_0} SEA_7) \oplus^W_{3t_0} SEA_{9.5})$$

Rotating Hyperplane. This data was used as a testbed for CVFDT versus VFDT in [14]. Examples for which $\sum_{i=1}^{d} w_i x_i \geq w_0$ are labeled positive, and examples for which $\sum_{i=1}^{d} w_i x_i < w_0$ are labeled negative. Hyperplanes are useful for simulating time-changing concepts, because we can change the orientation and position of the hyperplane in a smooth manner by changing the relative size of the weights.

Random RBF Generator. This generator was devised to offer an alternate complex concept type that is not straightforward to approximate with a decision tree model. The RBF (Radial Basis Function) generator works as follows: A fixed number of random centroids are generated. Each center has a random position, a single standard deviation, class label and weight. New examples are generated by selecting a center at random, taking weights into consideration so that centers with higher weight are more likely to be chosen. A random direction is chosen to offset the attribute values from the central point. Drift is introduced by moving the centroids with constant speed.

LED Generator. This data source originates from the CART book [5]. An implementation in C was donated to the UCI [1] machine learning repository by David Aha. The goal is to predict the digit displayed on a seven-segment LED display, where each attribute has a 10% chance of being inverted. The particular configuration of the generator used for the experiments (led) produces 24 binary attributes, 17 of which are irrelevant.

4.2 Real-World Data

The UCI machine learning repository [1] contains some real-world benchmark data for evaluating machine learning techniques. We consider three of the largest: Forest Covertype, Poker-Hand, and Electricity.

Forest Covertype. Contains the forest cover type for 30 x 30 meter cells obtained from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. It contains $581,012$ instances and 54 attributes, and it has been used in several papers on data stream classification [10,20].

Table 1. Comparison of Perceptron, Naive Bayes and Hoeffding Tree. The best individual accuracies are indicated in boldface.

	Perceptron			Naive Bayes			Hoeffding Tree		
	Time	Acc.	Mem.	Time	Acc.	Mem.	Time	Acc.	Mem.
RBF(0,0)	1.83	72.69 ± 0.28	0.01	4.15	72.04 ± 0.06	0.01	5.84	**86.75** ± 0.83	1.56
RBF(50,0.001)	2.67	**65.33** ± 0.04	0.01	5.02	53.23 ± 0.05	0.01	6.40	52.52 ± 0.17	1.50
RBF(10,0.001)	1.82	74.11 ± 0.09	0.01	4.14	75.79 ± 0.06	0.01	5.80	**83.72** ± 0.58	1.55
RBF(50,0.0001)	2.64	**69.34** ± 0.07	0.01	4.99	53.82 ± 0.04	0.01	7.04	55.78 ± 0.33	1.86
RBF(10,0.0001)	1.85	73.92 ± 0.16	0.01	4.14	75.18 ± 0.09	0.01	5.94	**83.59** ± 0.50	1.62
HYP(10,0.001)	1.55	**92.87** ± 0.43	0.01	3.91	77.64 ± 3.74	0.01	5.37	73.19 ± 2.60	1.60
HYP(10,0.0001)	1.56	**93.70** ± 0.03	0.01	3.90	90.23 ± 0.76	0.01	4.99	76.87 ± 1.46	1.40
SEA(50)	1.17	**87.15** ± 0.05	0.00	1.54	85.37 ± 0.00	0.00	2.54	85.68 ± 0.00	0.56
SEA(50000)	1.31	**86.85** ± 0.04	0.00	1.69	85.38 ± 0.00	0.00	2.71	85.59 ± 0.04	0.56
LED(50000)	6.64	**72.76** ± 0.01	0.02	8.97	54.02 ± 0.00	0.04	12.81	52.74 ± 0.15	4.98
CovType	12.21	**81.68**	0.05	22.81	60.52	0.08	13.43	68.30	2.59
Poker	5.36	3.34	0.01	9.25	59.55	0.02	5.46	**73.62**	1.11
Electricity	0.53	**79.07**	0.01	0.55	73.36	0.01	0.86	75.35	0.12
CovPokElec	20.87	13.69	0.06	56.52	24.34	0.11	42.82	**72.63**	10.03
	69.04 Acc. 0.12 RAM-Hours			67.18 Acc. 0.41 RAM-Hours			**73.31** Acc. 37 RAM-Hours		

Poker-Hand. Consists of 1,000,000 instances and 11 attributes. Each record of the Poker-Hand dataset is an example of a hand consisting of five playing cards drawn from a standard deck of 52. Each card is described using two attributes (suit and rank), for a total of 10 predictive attributes. There is one class attribute that describes the "Poker Hand".

Electricity. is another widely used dataset described by M. Harries [11] and analysed by Gama [9]. This data was collected from the Australian New South Wales Electricity Market. In this market, prices are not fixed and are affected by demand and supply of the market. They are set every five minutes. The ELEC dataset contains 45,312 instances. The class label identifies the change of the price relative to a moving average of the last 24 hours.

We use normalized versions of these datasets, so that the numerical values are between 0 and 1. With the Poker-Hand dataset, the cards are not ordered, i.e. a hand can be represented by any permutation, which makes it very hard for propositional learners, especially for linear ones. We use a modified version, where cards are sorted by rank and suit, and have removed duplicates. This dataset loses about 171,799 examples, and comes down to 829,201 examples.

These datasets are small compared to synthetic datasets we consider. Another important fact is that we do not know when drift occurs or indeed if there is any drift. We may simulate concept drift, joining the three datasets, merging attributes, and supposing that each dataset corresponds to a different concept

$$\text{CovPokElec} = (\text{CoverType} \oplus^{5,000}_{581,012} \text{Poker}) \oplus^{5,000}_{1,000,000} \text{ELEC}$$

As all examples need to have the same number of attributes, we simply concatenate all the attributes, and set the number of classes to the maximum number of classes of all the datasets.

4.3 Results

We use the datasets explained in the previous sections for evaluation. The experiments were performed on a 3 GHz Intel 64-bit machine with 2 GB of memory. The evaluation methodology used was Interleaved Test-Then-Train on 10 runs: every example was used for testing the model before using it to train. This interleaved test followed by train procedure was carried out on 10 million examples from the hyperplane and RandomRBF datasets, and one million examples from the SEA dataset. The parameters of these streams are the following:

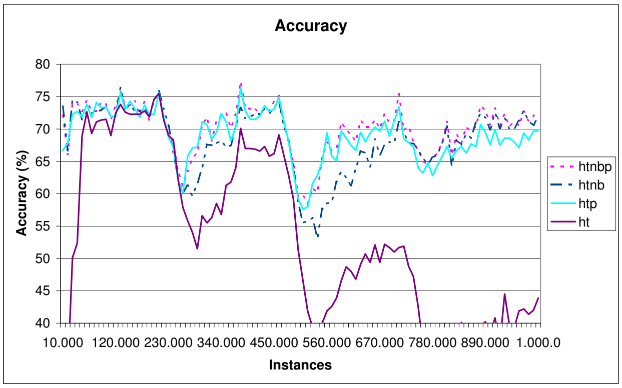
- RBF(x,v): RandomRBF data stream with x centroids moving at speed v.
- HYP(x,v): Hyperplane data stream with x attributes changing at speed v.
- SEA(v): SEA dataset, with length of change v.
- LED(v): LED dataset, with length of change v.

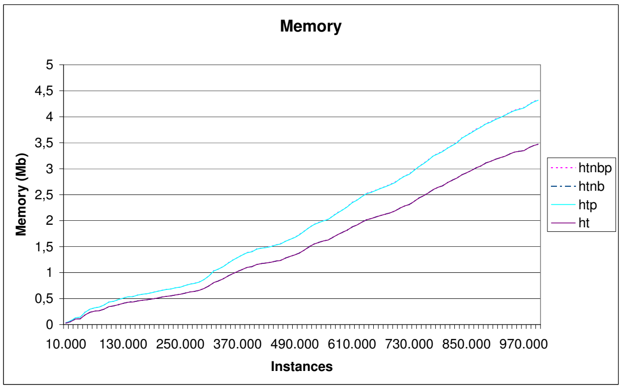
Tables 1, 2 and 3 report the final accuracy, and speed of the classification models induced on the synthetic data and the real datasets: Forest CoverType, Poker Hand, Electricity and CovPokElec. Accuracy is measured as the final percentage of examples correctly classified over the test/train interleaved evaluation. Time is measured in seconds, and memory in MB. The classification methods used are the following: perceptron, naive Bayes, Hoeffding Naive Bayes Tree (**hnbt**), Hoeffding Perceptron Tree (**hpt**), Hoeffding Naive Bayes Perceptron Tree (**hnbpt**), and `ADWIN` bagging using **hnbt**, **hpt**, and **hnbpt**.

The learning curves and model growth curves for the Led dataset are plotted in Figure 2. We observe that **ht** and **hpt** are the fastest decision trees. As the trees do not need more space to compute naive Bayes predictions at the leaves, **hnbt** uses the same memory as **ht**, and **hpnbt** uses the same memory as **hpt**. On accuracy, **ht** is the method that adapts more slowly to change, and during some time intervals **hpt** performs better than **hnbt**, but in other intervals performs worse. **hnbpt** is always the most or very close to the most accurate method as it is capable of making use of the decision of the majority class, naive Bayes and perceptron.

Table 1 shows the accuracy, speed and memory usage of a naive Bayes learner, a perceptron with $\eta = 1$ and a classic Hoeffding Tree with majority class learning at the leaves. As naive Bayes uses a Gaussian distribution to model numeric attributes, with different variances for each class, it does not yield a linear separator as the perceptron does. In general terms, we see that the perceptron and the Hoeffding Tree are the fastest methods, but the Hoeffding Tree needs more memory. Comparing using RAM-Hours, naive Bayes needs 3.5 times more RAM-Hours than the perceptron, and the Hoeffding Tree needs 89 more RAM-Hours than naive Bayes. Note that using $\eta = 1$ we obtain a very fast adaptive method, but for some datasets like Poker, the results are worse than obtained using a more conservative rate like $\eta = 0.01$. Choosing an optimal η remains an open problem for further research.

Table 2 shows, for the Hoeffding tree models, their accuracy, speed and memory. We see that **hpt** is much faster than **hnbt**, and more accurate in several streams. **hnbpt** is more accurate than **hnbt** and **hpt**, but it needs more time.





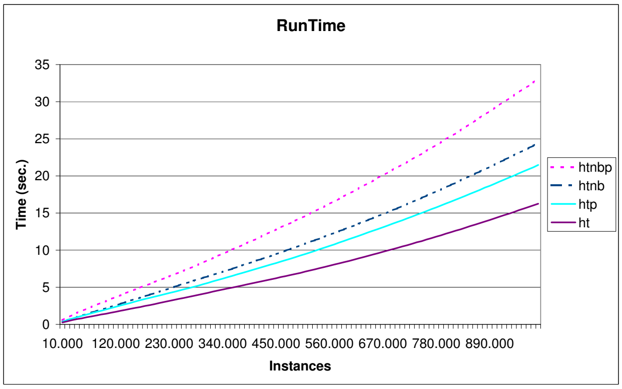


Fig. 2. Accuracy, runtime and memory on the LED data with three concept drifts

Table 2. Comparison of **hnbt**, **hpt**, and **hnbpt** algorithms. The best individual accuracies are indicated in boldface.

	hnbt			**hpt**			**hnbpt**		
	Time	Acc.	Mem.	Time	Acc.	Mem.	Time	Acc.	Mem.
RBF(0,0)	9.03	90.78 ± 0.46	1.57	8.04	90.33 ± 0.49	2.30	10.77	**91.07** ± 0.44	2.30
RBF(50,0.001)	10.98	57.70 ± 0.22	1.51	8.66	**68.95** ± 0.31	2.20	11.32	68.65 ± 0.32	2.20
RBF(10,0.001)	9.07	87.24 ± 0.29	1.56	8.03	87.61 ± 0.36	2.28	10.77	**87.98** ± 0.32	2.28
RBF(50,0.0001)	11.62	69.00 ± 0.46	1.86	9.54	**79.91** ± 0.42	2.72	12.45	79.88 ± 0.39	2.72
RBF(10,0.0001)	9.28	88.47 ± 0.37	1.63	8.20	89.37 ± 0.32	2.38	10.98	**89.74** ± 0.33	2.39
HYP(10,0.001)	9.73	83.24 ± 2.29	1.61	7.57	82.74 ± 1.13	2.34	11.45	**84.54** ± 1.40	2.34
HYP(10,0.0001)	9.37	**88.42** ± 0.36	1.40	7.08	82.59 ± 0.62	2.04	11.30	88.40 ± 0.36	2.04
SEA(50)	3.70	86.63 ± 0.00	0.57	4.65	86.73 ± 0.01	1.23	5.49	**87.41** ± 0.01	1.24
SEA(50000)	4.51	86.44 ± 0.03	0.57	4.85	86.41 ± 0.07	1.23	5.64	**87.12** ± 0.07	1.24
LED(50000)	21.28	68.06 ± 0.10	4.99	18.64	68.87 ± 0.07	6.00	24.99	**70.04** ± 0.03	6.00
CovType	24.73	81.06	2.59	16.53	83.59	3.46	22.16	**85.77**	3.46
Poker	9.81	**83.05**	1.12	8.40	74.02	1.82	11.40	82.93	1.82
Electricity	0.96	80.69	0.12	0.93	84.24	0.21	1.07	**84.34**	0.21
CovPokElec	68.37	**83.41**	10.05	49.37	73.33	13.53	69.70	83.28	13.53
	81.01 Acc. 61.58 RAM-Hours			81.33 Acc. 68.56 RAM-Hours			**83.65** Acc. 93.84 RAM-Hours		

Table 3. Comparison of `ADWIN` bagging with **hnbt**, **hpt**, and **hnbpt**. The best individual accuracies are indicated in boldface.

	`ADWIN` Bagging **hnbt**			`ADWIN` Bagging **hpt**			`ADWIN` Bagging **hnbpt**		
	Time	Acc.	Mem	Time	Acc.	Mem	Time	Acc.	Mem
RBF(0,0)	102.22	94.30 ± 0.07	16.22	88.84	93.70 ± 0.10	23.35	115.31	**94.36** ± 0.07	23.98
RBF(50,0.001)	88.75	67.67 ± 0.16	0.12	41.15	**74.82** ± 0.26	2.27	76.34	74.13 ± 0.40	2.84
RBF(10,0.001)	97.65	89.74 ± 0.09	12.97	80.65	90.36 ± 0.11	20.44	106.64	**90.55** ± 0.10	19.73
RBF(50,0.0001)	90.64	84.99 ± 0.17	1.33	61.82	87.24 ± 0.14	9.66	94.22	**87.97** ± 0.11	9.39
RBF(10,0.0001)	100.21	91.97 ± 0.07	13.81	82.54	92.53 ± 0.12	21.56	109.59	**93.01** ± 0.05	21.32
HYP(10,0.001)	90.77	89.92 ± 0.31	3.02	31.86	91.20 ± 0.99	1.11	69.23	**91.45** ± 0.81	2.43
HYP(10,0.0001)	107.36	91.30 ± 0.21	8.22	30.77	**93.63** ± 0.21	0.08	50.57	93.61 ± 0.24	0.11
SEA(50)	44.48	88.22 ± 0.22	4.33	54.09	88.08 ± 0.07	11.07	59.19	**88.60** ± 0.09	10.17
SEA(50000)	41.03	88.61 ± 0.07	2.69	53.44	87.85 ± 0.07	10.78	54.07	**88.65** ± 0.05	6.83
LED(50000)	150.62	**73.14** ± 0.02	5.09	93.09	72.82 ± 0.03	14.84	130.56	73.02 ± 0.02	8.45
CovType	165.75	85.73	0.80	50.06	86.33	1.66	115.58	**87.88**	1.25
Poker	57.40	**74.56**	0.09	37.14	65.76	0.21	73.41	74.36	0.16
Electricity	3.17	84.36	0.13	2.59	85.22	0.44	3.55	**86.44**	0.30
CovPokElec	363.70	**78.96**	1.18	118.64	67.02	1.13	402.20	78.77	1.54
	84.53 Acc. 1028.02 RAM-Hours			84.04 Acc. 957.38 RAM-Hours			**85.91** Acc. 1547.33 RAM-Hours		

Comparing RAM-Hours, **hpt** needs 1.11 times more RAM-Hours than **hnbt**, and 2.54 more RAM-Hours than **ht**, and **hnbpt** needs 1.37 more than **hpt**.

Table 3 reports the accuracy, speed and memory of `ADWIN` bagging using **hnbt**, **hpt**, and **hnbpt**. `ADWIN` bagging using **hnbpt** is the most accurate method, but it uses more time and memory than the other variants. In RAM-Hours, it needs 1.62 times more than `ADWIN` bagging using **hpt**, and 1.51 times more than `ADWIN` bagging using **hnbt**. `ADWIN` bagging using **hpt** needs fewer resources than `ADWIN` bagging using **hnbt**.

Comparing **hnbpt** from Table 2 with the single perceptron from Table 1, we obtain a 20% better accuracy, but at a cost of 780 times the amount of RAM-Hours. Comparing `ADWIN` bagging using **hnbpt** (Table 3) with a single **hnbpt** (Table 2) we obtain a 3% better accuracy, at 16.50 times the RAM-Hours.

Concept drift is handled well by the proposed `ADWIN` bagging algorithms, excluding the poor performance of the **hpt**-based classifier on CovPokElec, which is due to the nature of the Poker dataset. Decision trees alone do not deal as well with evolving streaming data, as they have limited capability of adaption.

A tradeoff between RAM-Hours and accuracy could be to use single perceptrons when resources are scarce, and ADWIN bagging methods when more accuracy is needed. Note that to gain an increase of 24% in accuracy, we have to increase by more than $10,000$ times the RAM-Hours needed; this is the difference of magnitude between the RAM-Hours needed for a single perceptron and for a more accurate ADWIN bagging method.

5 Conclusions

We have investigated four perceptron-based methods for data streams: a single learner, a decision tree, a hybrid tree, and an ensemble method. These methods use perceptrons with a sigmoid activation function, optimizing the squared error, with one perceptron per class value. We observe that perceptron-based methods are competitive in accuracy and use of resources. We have introduced the use of RAM-Hours as a performance measure. Using RAM-Hours, it is easy to compare resources used by classifier algorithms.

As future work, we would like to build new methods based on the perceptron, with an adaptive learning rate. We think that in changing scenarios, using a flexible learning rate may allow us to obtain more accurate methods, without incurring large additional runtime or memory costs.

References

1. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
2. Bennett, K., Cristianini, N., Shawe-Taylor, J., Wu, D.: Enlarging the margins in perceptron decision trees. Machine Learning 41(3), 295–313 (2000)
3. Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: SDM (2007)
4. Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavaldà, R.: New ensemble methods for evolving data streams. In: KDD, pp. 139–148 (2009)
5. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth, Belmont (1984)
6. Domingos, P., Hulten, G.: Mining high-speed data streams. In: KDD, pp. 71–80 (2000)
7. Frank, E., Wang, Y., Inglis, S., Holmes, G., Witten, I.H.: Using model trees for classification. Machine Learning 32(1), 63–76 (1998)
8. Gama, J.: On Combining Classification Algorithms. VDM Verlag (2009)
9. Gama, J., Medas, P., Castillo, G., Rodrigues, P.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)
10. Gama, J., Rocha, R., Medas, P.: Accurate decision trees for mining high-speed data streams. In: KDD, pp. 523–528 (2003)
11. Harries, M.: Splice-2 comparative evaluation: Electricity pricing. Technical report, The University of South Wales (1999)

12. Holmes, G., Kirkby, R., Pfahringer, B.: Stress-testing Hoeffding trees. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 495–502. Springer, Heidelberg (2005)
13. Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive Online Analysis (2007), http://sourceforge.net/projects/moa-datastream
14. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: KDD, pp. 97–106 (2001)
15. Ikonomovska, E., Gama, J.: Learning model trees from data streams. Discovery Science, 52–63 (2008)
16. Ikonomovska, E., Gama, J., Sebastião, R., Gjorgjevik, D.: Regression trees from data streams with drift detection. In: Gama, J., Costa, V.S., Jorge, A.M., Brazdil, P.B. (eds.) DS 2009. LNCS, vol. 5808, pp. 121–135. Springer, Heidelberg (2009)
17. Landwehr, N., Hall, M., Frank, E.: Logistic model trees. Machine Learning 59(1-2), 161–205 (2005)
18. Murthy, S.K.: Automatic construction of decision trees from data: A multi-disciplinary survey. Data Min. Knowl. Discov. 2(4), 345–389 (1998)
19. Oza, N., Russell, S.: Online bagging and boosting. In: Artificial Intelligence and Statistics 2001, pp. 105–112. Morgan Kaufmann, San Francisco (2001)
20. Oza, N.C., Russell, S.J.: Experimental comparisons of online and batch versions of bagging and boosting. In: KDD, pp. 359–364 (2001)
21. Safavian, S.R., Landgrebe, D.: A survey of decision tree classifier methodology. IEEE Transactions on Systems, Man and Cybernetics 21(3), 660–674 (1991)
22. Schlimmer, J.C., Fisher, D.H.: A case study of incremental concept induction. In: AAAI, pp. 496–501 (1986)
23. Street, W.N., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: KDD, pp. 377–382 (2001)
24. Utgoff, P.E.: Perceptron trees: A case study in hybrid concept representations. In: AAAI, pp. 601–606 (1988)
25. Velte, T., Velte, A., Elsenpeter, R.: Cloud Computing, A Practical Approach. McGraw-Hill, Inc., New York (2010)
26. Zhou, Z., Chen, Z.: Hybrid decision tree. Knowledge-based systems 15(8), 515–528 (2002)

Classification and Novel Class Detection in Data Streams with Active Mining

Mohammad M. Masud[1], Jing Gao[2], Latifur Khan[1], Jiawei Han[2], and Bhavani Thuraisingham[1]

[1] Department of Computer Science, University of Texas at Dallas
[2] Department of Computer Science, University of Illinois at Urbana-Champaign

Abstract. We present ActMiner, which addresses four major challenges to data stream classification, namely, infinite length, concept-drift, concept-evolution, and limited labeled data. Most of the existing data stream classification techniques address only the infinite length and concept-drift problems. Our previous work, MineClass, addresses the concept-evolution problem in addition to addressing the infinite length and concept-drift problems. Concept-evolution occurs in the stream when novel classes arrive. However, most of the existing data stream classification techniques, including MineClass, require that all the instances in a data stream be labeled by human experts and become available for training. This assumption is impractical, since data labeling is both time consuming and costly. Therefore, it is impossible to label a majority of the data points in a high-speed data stream. This scarcity of labeled data naturally leads to poorly trained classifiers. ActMiner actively selects only those data points for labeling for which the expected classification error is high. Therefore, ActMiner extends MineClass, and addresses the limited labeled data problem in addition to addressing the other three problems. It outperforms the state-of-the-art data stream classification techniques that use ten times or more labeled data than ActMiner.

1 Introduction

Data stream classification is more challenging than classifying static data because of several unique properties of data streams. First, data streams are assumed to have *infinite length*, which makes it impractical to store and use all the historical data for training. Therefore, traditional multi-pass learning algorithms are not directly applicable to data streams. Second, data streams observe *concept-drift*, which occurs when the underlying concept of the data changes over time. In order to address concept-drift, a classification model must continuously adapt itself to the most recent concept. Third, data streams also observe *concept-evolution*, which occurs when a novel class appears in the stream. In order to cope with concept-evolution, a classification model must be able to automatically detect novel classes when they appear, before being trained with the labeled instances of the novel class. Finally, high speed data streams suffer from *insufficient labeled data*. This is because, manual labeling is both costly and time consuming. Therefore, the speed at which the data points are labeled lags far behind the speed at which data points arrive in the stream, leaving most of the data points in the stream as unlabeled. So, supervised classification techniques suffer from the scarcity of labeled data for learning, resulting in a poorly built classifier. Most existing data stream classification techniques address only the infinite length, and concept-drift problems [1–3]. Our previous work MineClass [4] addresses the concept-evolution problem in addition to the infinite length and concept-drift problems. However, it did not address the limited labeled data problem. Our current work, ActMiner, extends MineClass by addressing all the four problems and providing a more realistic data stream classification framework than the state-of-the-art.

A solution to the infinite length problem is incremental learning, which requires a single pass over the training data. In order to cope with concept-drift, a classifier must be continuously updated to be consistent with the most recent concept. ActMiner applies a hybrid batch-incremental process [2, 5] to solve the infinite length and concept-drift problems. It divides the data stream into equal sized chunks and trains a classification model from each chunk. An ensemble of M such models is used to classify the unlabeled data. When a new data chunk becomes available for training, a new model is trained, and an old model from the ensemble is replaced with the new model. The victim for the replacement is chosen by evaluating the accuracy of each model on the latest labeled chunk. In this way, the ensemble is kept up-to-date. ActMiner also solves the concept-evolution problem by automatically detecting novel classes in the data stream. In order to detect novel class, it first identifies the test instances that are *well-separated* from the training data, and tag them as *Raw outlier*. Then raw outliers that possibly appear as a result of concept-drift or noise are filtered out. If a sufficient number of such strongly cohesive filtered outliers (called *F-outlier*s) are observed, a novel class is assumed to have appeared, and the *F-outlier*s are classified as novel class instances. Finally, ActMiner solves the limited labeled data problem by requiring only a few selected instances to be labeled. It identifies the instances for which the classification model has the highest expected error. This selection is done without knowing the true labels of those instances. By selecting only a few instances for labeling, it saves 90% or more labeling time and cost, than traditional approaches that require all instances to be labeled.

We have several contributions. First, we propose a framework that addresses four major challenges in data stream classification. To the best of our knowledge, no other existing data stream classification technique addresses all these four problems in a single framework. Second, we show how to select only a few instances in the stream for labeling, and justify this selection process both theoretically and empirically. Finally, our technique outperforms state-of-the-art data stream classification techniques using ten times or even less amount of labeled data for training. The rest of the paper is organized as follows. Section 2 discusses the related works in data stream classification. Section 3 describes the proposed approach. Section 4 then presents the experiments and analyzes the results. Section 5 concludes with directions to future work.

2 Related Work

Related works in data stream classification can be divided into three groups: i) approaches that address the infinite length and concept-drift problems, ii) approaches that address the infinite length, concept-drift, and limited labeled data problems, and iii) approaches that address the infinite length, concept-drift, and concept-evolution problems. Groups i) and ii) again can be subdivided into two subgroups: single model and ensemble classification approach.

Most of the existing techniques fall into group i). The single-model approaches in group i) apply incremental learning and adapt themselves to the most recent concept by continuously updating the current model to accommodate concept drift [1, 3, 6]. Ensemble techniques [2, 5] maintain an ensemble of models, and use ensemble voting to classify unlabeled instances. These techniques address the infinite length problem by keeping a fixed-size ensemble, and address the concept-drift problem by updating the ensemble with newer models. ActMiner also applies an ensemble classification technique. Techniques in group ii) goes one step ahead of group i) by addressing the limited labeled data problem. Some of them apply active learning [7, 8] to select the instances to be labeled, and some [9] apply random sampling along with semi-supervised clustering. ActMiner also applies active learning, but its data selection process is different from the others. Unlike other active mining techniques such as [7] that requires extra computational overhead to select the data, ActMiner does the selection on the fly during classification. Moreover, none of these approaches address the concept-evolution problem, but ActMiner does.

Techniques in group iii) are the most rare. An unsupervised novel concept detection technique for data streams is proposed in [10], but it is not applicable to multi-class classification. Our previous work MineClass [4] addresses the concept-evolution problem on a multi-class classification framework. It can detect the arrival of a novel class automatically, without being trained with any labeled instances of that class. However, it does not address the limited labeled data problem, and requires that all instances in the stream be labeled and available for training. ActMiner extends MineClass by requiring only a few chosen instances to be labeled, thereby reducing the labeling cost by 90% or more.

3 ActMiner: Active Classification and Novel Class Detection

In this section we discuss ActMiner in details. Before describing ActMiner, we briefly introduce MineClass, and present some definitions.

3.1 Background: Novel Class Detection with MineClass

ActMiner is based on our previous work MineClass [4], which also does data stream classification and novel class detection. MineClass is an ensemble classification approach, which keeps an ensemble L of M classification models, i.e., L=$\{L_1,...,L_M\}$. First, we define the concept of *novel class* and *existing class.*

Definition 1 (Existing class and Novel class). *Let L be the current ensemble of classification models. A class c is an existing class if at least one of the models $L_i \in L$ has been trained with class c. Otherwise, c is a novel class.*

The basic assumption in novel class detection lies in the following property.

Property 1. *Let x be an arbitrary instance belonging to a class c', and c be any class other than c'. Also, let $\lambda_{c',q}(x)$ be the q-nearest neighbors of x within class c', and $\lambda_{c,q}(x)$ be the q-nearest neighbors of x within class c. Then the mean distance from x to $\lambda_{c',q}(x)$ is less than the mean distance from x to $\lambda_{c,q}(x)$, for any class $c \neq c'$.*

In other words, property 1 states that an instance is closer to other same class instances and farther from the instances of any other class. Therefore, if a novel class arrives, the instances belonging to that class must be closer to other novel class instances and far from any existing class instances. This is the basic idea in detecting novel class with MineClass. MineClass detects novel classes in three steps: i) creating decision boundary for a classifier during its training, ii) detecting and filtering outliers, and iii) computing cohesion among the outliers, and separation of the outliers from the training data.

The decision boundaries are created by clustering the training data, and saving the cluster centroids and radii as pseudopoints. Each pseudopoint represents a hypersphere in the feature space. Union of all the hyperspheres in a classification model constitutes the decision boundary for that model. The decision boundary for the ensemble of models is the union of the decision boundaries of each model in the ensemble. Any test instance falling outside the decision boundary of the ensemble of models is considered an outlier, called *F-outlier*.

Definition 2 (F$-$outlier). *A test instance is an F$-$outlier (i.e., filtered outlier) if it is outside the decision boundary of* all *classifiers $L_i \in L$.*

If any test instance x is inside the decision boundary, then it can be shown that there is at least one existing class instance x', such that the mean distance from x to the existing class instances is less than the mean distance from x' to the existing class instances. Therefore, according to property 1, x must be an existing class instance. Any *F-outlier* is a potential novel class instance, because it is outside the decision boundary of the ensemble of models, and therefore, we its membership in the existing classes cannot be guaranteed. However, only one *F-outlier* does not imply a novel class. We need to know whether there are enough *F-outlier*s that are sufficiently close to each other and far from the existing class instances. This is done by computing the cohesion among *F-outlier*s and separation of *F-outlier*s from existing class instances. This is done using the following equation: $q\text{-}NSC(x) = \frac{b_{min}(x) - a(x)}{max(b_{min}(x), a(x))}$, where x is an *F-outlier*, $b_{min}(x)$ is the mean distance from x to its q-nearest existing class instances and $a(x)$ is the mean distance from x to its q-nearest *F-outlier* instances. A positive value indicates that x is closer to other *F-outlier* instances than the existing class instances. If q-NSC(x) is positive for at least q *F-outlier* instances, then a novel class is assumed to have arrived. This is the basic working principle of the DetectNovelClass() function in algorithm 1.

3.2 ActMiner Algorithm

ActMiner, which stands for Active Classifier for Data Streams with novel class Miner, performs classification and novel class detection in data streams while requiring very small amount of labeled data for training. The top level algorithm is sketched in algorithm 1.

Algorithm 1. ActMiner

```
1: $L \leftarrow$ Build-initial-ensemble(), $\mathcal{L} \leftarrow$ empty //training data
2: **while true do**
3:    $D_n \leftarrow$ the latest data chunk in the stream
4:    //Classification, outlier detection, novel class detection
5:    $buf \leftarrow$ empty //temporary buffer
6:    **for each** $x_k \in D_n$ **do**
7:       $< fout, \hat{y}_k > \leftarrow$ Classify($x_k$,$L$) //$\hat{y}_k$ is the predicted class label of $x_k$
8:       **if** $fout$ = **true then** $buf \Leftarrow x_k$ //enqueue into buffer
9:       **else** output prediction $< x_k, \hat{y}_k >$ **end if**
10:   **end for**
11:   $found \leftarrow$ DetectNovelClass($L$,$buf$) //(see section 3.1)
12:   **if** $found$=**true then**
13:      **for each** novel class instance $x_k \in buf$ **do** $\hat{y}_k \leftarrow$ “novel class” **end for**
14:   **end if**
15:   **for each** instance $x_k \in buf$ output prediction $< x_k, \hat{y}_k >$ **end for**
16:   //Label the chunk
17:   **for each** $x_k \in D_n$ **do**
18:      **if** $x_k$ is an weakly classified instance (WCI)
19:         **then** $\mathcal{L} \Leftarrow < x_k, y_k >$ //label it and save ($y_k$ is the true class label of $x_k$)
20:         **else** $\mathcal{L} \Leftarrow < x_k, \hat{y}_k >$ //save in training buffer with the predicted class label
21:      **end if**
22:   **end for**
23:   //Training
24:   $L' \leftarrow$ **Train-and-save-decision-boundary** ($\mathcal{L}$) //(see section 3.1)
25:   $L \leftarrow$ **Update**($L$,$L'$,$\mathcal{L}$)
26:   $\mathcal{L} \leftarrow$ empty
27: **end while**
```

The algorithm starts with building the initial ensemble $L = \{L_1, ..., L_M\}$ with the first few data chunks of the stream (line 1), and initializing the training buffer. Then a **while** loop (line 2) runs indefinitely until the stream is finished. Within the while loop, the latest data chunk D_n is examined. Each instance x_k in D_n is first passed to the Classify() function, which uses the existing ensemble to get its predicted class $\hat{y}_k$ and its F-*outlier* status (line 7). If it is identified as an F-*outlier*, then it is temporarily saved in a buffer buf for further inspection (line 8), otherwise, we output its predicted class (line 9). Then we call the DetectNovelClass() function to inspect buf to detect whether any novel class has arrived (line 11). If a novel class has arrived then the novel class instances are classified as “novel class” (line 13). Then the class predictions of all instances in *buf* are sent to the output (line 15). We then select the instances that need to be labeled (lines 17-22). Only the instances identified as *Weakly Classified Instance* (WCI) are required to be labeled by human experts, and they are saved in the training buffer with their *true* class labels (line 19). We will explain WCI shortly. All other instances remain as unlabeled, and they are saved in the training buffer with their *predicted* class labels (line 20). A new model L' is trained with the training buffer (line 24), and this model is used to update the existing ensemble L (line 25). Updating is done by first evaluating each model $L_i \in L$ on $\mathcal{L}$, and replacing the worst (based on accuracy) of them with L'. ActMiner can be applied to any base learning algorithm in general. The only operation that needs to be specific to a learning algorithm is *train and save decision boundary.*

3.3 Data Selection for Labeling

Unlike MineClass, ActMiner does not need all the instances in the training data to have true labels. Only those instances need to be labeled about whose class labels MineClass is the most uncertain. We call these instances as "weakly classified instances" or WCIs. ActMiner finds the WCIs and presents them to the user for labeling, because the ensemble has the highest uncertainty in classifying the WCIs. In order to perform ensemble voting on an instance x_j, first we initialize a vector $V = \{v[1], ..., v[C]\}$ to zeros, where C is the total number of classes, and each $v[k]$ represents a real value. Let classifier L_i predicts the class label of x_j to be c, where $c \in \{1, ..., C\}$. Then we increment $v[c]$ by 1. Let $v[max]$ represent the maximum among all $v[i]$. Then the predicted class of x_j is max. An instance x_j is a WCI if either i) The instance has been identified as an *F-outlier* (see definition 2), or ii) The *ratio* of its majority vote to its total vote is less than the *Minimum Majority Threshold* (MMT), a user-defined parameter.

For condition i), consider that *F-outlier*s are outside the decision boundary of all the models in the ensemble. So the ensemble has the highest uncertainty in classifying them. Therefore, *F-outlier*s are considered as WCIs and need to be labeled. For condition ii), let us denote the ratio with *Majority to Sum* (M2S) ratio. Let $v[max]$ be maximum in the vector V, and let $s = \sum_{i=1}^{C} v[i]$. Therefore, the M2S ratio of x_j is given by: $M2S(x_j) = \frac{v[max]}{s}$. The data point x_j is considered to be a WCI if M2S$(x_j) <$ MMT. A lower value of M2S(x_j) indicates higher uncertainty in classifying that instance, and vice versa.

Next we justify the reason for labeling the WCIs of the second type, i.e., instances that have M2S$(x_j) <$ MMT. We show that the ensemble classification error is higher for the instances having lower M2S.

Lemma 1. *Let $\mathcal{A}$ and $\mathcal{B}$ be two sets of disjoint datapoints such that for any $x_a \in \mathcal{A}$, and $x_b \in \mathcal{B}$, M2S$(x_a) <$ M2S(x_b). Then the ensemble error on $\mathcal{A}$ is higher than the ensemble error on $\mathcal{B}$.*

Proof. Given an instance x, the posterior probability distribution of class c is $p(c|x)$. Let C be the total number of classes, and $c \in \{1, ..., C\}$. According to

Tumer and Ghosh [11], a classifier is trained to learn a function $f_c(.)$ that approximates this posterior probability (i.e., probability of classifying x into class c): $f_c(x) = p(c|x) + \eta_c(x)$ where $\eta_c(x)$ is the error of $f_c(x)$ relative to $p(c|x)$. This is the error in addition to Bayes error and usually referred to as the *added error*. This error occurs either due to the bias of the learning algorithm, and/or the variance of the learned model. According to [11], the expected added error can be obtained from the following formula: $Error = \frac{\sigma^2_{\eta_c(x)}}{s}$ where $\sigma^2_{\eta_c(x)}$ is the variance of $\eta_c(x)$, and s is the difference between the derivatives of $p(c|x)$ and $p(\neg c|x)$, which is independent of the learned classifier.

Let $L = \{L_1, ..., L_M\}$ be an ensemble of M classifiers, where each classifier L_i is trained from a data chunk. If we average the outputs of the classifiers in a M-classifier ensemble, then according to [11], the probability of the ensemble in classifying x into class c is: $f_c^{avg}(x) = \frac{1}{M}\sum_{m=1}^{M} f_c^m(x) = p(c|x) + \eta_c^{avg}(x)$, where $f_c^{avg}(x)$ is the output of the ensemble L, $f_c^m(x)$ is the output of the m-th classifier L_m, and $\eta_c^{avg}(x)$ is the added error of the ensemble, given by: $\eta_c^{avg}(x) = \frac{1}{M}\sum_{m=1}^{M} \eta_c^m(x)$, where $\eta_c^m(x)$ is the added error of the m-th classifier in the ensemble. Assuming the error variances are independent, the variance of $\eta_c^{avg}(x)$, i.e., the error variance of the ensemble, $\sigma^2_{\eta_c^{avg}(x)}$, is given by:

$$\sigma^2_{\eta_c^{avg}(x)} = \frac{1}{M^2}\sum_{m=1}^{M} \sigma^2_{\eta_c^m(x)}, \tag{1}$$

where $\sigma^2_{\eta_c^m(x)}$ is the variance of $\eta_c^m(x)$.

Also, let $\sigma^2_{\eta_c^{avg}(x_a)}(\mathcal{A})$, and $\sigma^2_{\eta_c^{avg}(x_b)}(\mathcal{B})$ be the variances of the ensemble error on $\mathcal{A}$, and $\mathcal{B}$, respectively. Let $z_c(x)$ be 1 if the true class label of x is c, and $z_c(x)$ be 0, otherwise. Also, let $f_c^m(x)$ be either 0 or 1. The error variance of classifier L_m on $\mathcal{A}$ is given by [7]:

$$\sigma^2_{\eta_c^m(x_a)}(\mathcal{A}) = \frac{1}{|\mathcal{A}|}\sum_{x_a \in \mathcal{A}} (z_c(x_a) - f_c^m(x_a))^2, \tag{2}$$

where $(z_c(x_a) - f_c^m(x_a))^2$ is the squared error of classifier L_m on instance x_a. Since we assume that $f_c^m(x_a)$ is either 0 or 1, it follows that $(z_c(x_a) - f_c^m(x_a))^2 = 0$ if the prediction of L_m is correct, and $= 1$, otherwise. Let x_a be an arbitrary instance in $\mathcal{A}$, and let $r(x_a)$ be the majority vote count of x_a. Also, let us divide the classifiers into two groups. Let *group 1* be $\{L_{m_j}\}_{j=1}^{r(x_a)}$, i.e., the classifiers that contributed to the majority vote, and *group 2* be $\{L_{m_j}\}_{j=r(x_a)+1}^{M}$, i.e., all other classifiers. Since we consider that the errors of the classifiers are independent, it is highly unlikely that majority of the classifiers will make the same mistake. Therefore, we may consider the votes in favor of the majority class to be correct. So, all classifiers in group 1 has correct prediction, and all other classifiers have incorrect predictions. The *combined squared error* (CSE) of the *individual* classifiers in classifying x_a into class c is:

$$\sum_{m=1}^{M}(z_c(x_a) - f_c^m(x_a))^2 = \sum_{j=1}^{r(x_a)}(z_c(x_a) - f_c^{m_j}(x_a))^2 + \sum_{j=r(x_a)+1}^{M}(z_c(x_a) - f_c^{m_j}(x_a))^2$$

$$= 0 + \sum_{j=r(x_a)+1}^{M}(z_c(x_a) - f_c^{m_j}(x_a))^2 \tag{3}$$

Note that CSE is the sum of the squared errors of individual classifiers in the ensemble, not the error of the ensemble itself. Also, note that each component of group 2 in the CSE, i,e,. each $(z_c(x_a) - f_c^{m_j}(x_a))^2$, $j > r(x_a)$ contributes 1 to the sum (since the prediction is wrong). Now we may proceed as follows:

$$M2S(x_a) < M2S(x_b) \Rightarrow r(x_a) < r(x_b) \quad \text{(since the total vote} = M) \tag{4}$$

This implies that the size of group 2 for x_a is larger than that for x_b. Therefore, the CSE in classifying x_a is greater than that of x_b, since each component of group 2 in CSE contributes 1 to the sum. Continuing from eqn (4),

$$\Rightarrow \sum_{j=r(x_a)+1}^{M}(z_c(x_a) - f_c^{m_j}(x_a))^2 > \sum_{j=r(x_b)+1}^{M}(z_c(x_b) - f_c^{m_j}(x_b))^2$$

$$\Rightarrow \sum_{m=1}^{M}(z_c(x_a) - f_c^m(x_a))^2 > \sum_{m=1}^{M}(z_c(x_b) - f_c^m(x_b))^2 \quad \text{(using eqn 3)} \tag{5}$$

Now, according to the Lemma statement, for any pair $(x_a \in \mathcal{A}, x_b \in \mathcal{B})$, $M2S(x_a) < M2S(x_b)$ holds, and hence, inequality (5) holds. Therefore, the mean CSE of set $\mathcal{A}$ must be less than the mean CSE of set $\mathcal{B}$, i.e.,

$$\Rightarrow \frac{1}{|\mathcal{A}|}\sum_{x_a\in\mathcal{A}}\sum_{m=1}^{M}(z_c(x_a) - f_c^m(x_a))^2 > \frac{1}{|\mathcal{B}|}\sum_{x_b\in\mathcal{B}}\sum_{m=1}^{M}(z_c(x_b) - f_c^m(x_b))^2$$

$$\Rightarrow \sum_{m=1}^{M}(\frac{1}{|\mathcal{A}|}\sum_{x_a\in\mathcal{A}}(z_c(x_a) - f_c^m(x_a))^2) > \sum_{m=1}^{M}(\frac{1}{|\mathcal{B}|}\sum_{x_b\in\mathcal{B}}(z_c(x_b) - f_c^m(x_b))^2)$$

$$\Rightarrow \sum_{m=1}^{M}\sigma^2_{\eta_c^m(x_a)}(\mathcal{A}) > \sum_{m=1}^{M}\sigma^2_{\eta_c^m(x_b)}(\mathcal{B}) \quad \text{(using eqn 2)}$$

$$\Rightarrow \sigma^2_{\eta_c^{avg}(x_a)}(\mathcal{A}) > \sigma^2_{\eta_c^{avg}(x_b)}(\mathcal{B}) \quad \text{(using eqn 1)}$$

That is, the ensemble error variance, and hence, the ensemble error (since error variance is proportional to error) on $\mathcal{A}$ is higher than that of $\mathcal{B}$. □

4 Experiments

In this section we describe the datasets, experimental environment, and discuss and analyze the results.

4.1 Data Sets and Experimental Setup

We use two synthetic and two real datasets for evaluation. These are: Synthetic data with only concept-drift **(SynC)**, Synthetic data with concept-drift and novel-class **(SynCN)**, Real data - KDDCup 99 network intrusion detection **(KDD)**, and Real data - Forest cover dataset from UCI repository **(Forest)**. Due to space limitation, we omit the details of the datasets. Details can be found in [4]. We use the following parameter settings, unless mentioned otherwise: i) K (number of pseudopoints per classifier) = 50, ii) q (minimum number of instances required to declare novel class) = 50, iii) L (ensemble size) = 6, iv) S (chunk size) = 2,000. v) MMT (minimum majority threshold) = 0.5.

4.2 Baseline Approach

We use the same baseline techniques that were used to compare with MineClass [4]. Since to the best of our knowledge, there is no technique that can both classify and detect novel class in data streams, a combination of two baseline techniques are used in MineClass: $OLINDDA$ [10], and Weighted Classifier Ensemble (WCE) [2], where the former works as novel class detector, and the latter performs classification. For each chunk, we first detect the novel class instances using $OLINDDA$. All other instances in the chunk are assumed to be in the existing classes, and they are classified using WCE. We use $OLINDDA$ as the novelty detector, since it is a recently proposed algorithm that is shown to have outperformed other novelty detection techniques in data streams [10].

However, $OLINDDA$ assumes that there is only one "normal" class, and all other classes are "novel". So, it is not directly applicable to the multi-class novelty detection problem, where any combination of classes can be considered as the "existing" classes. We propose two alternative solutions. First, we build parallel $OLINDDA$ models, one for each class, which evolve simultaneously. Whenever the instances of a novel class appear, we create a new $OLINDDA$ model for that class. A test instance is declared as novel, if *all the existing class models* identify this instance as novel. We will refer to this baseline method as WCE-OLINDDA_PARALLEL. Second, we initially build an $OLINDDA$ model with all the available classes. Whenever a novel class is found, the class is absorbed into the existing $OLINDDA$ model. Thus, only one "normal" model is maintained throughout the stream. This will be referred to as WCE-OLINDDA_SINGLE. In all experiments, the ensemble size and chunk-size are kept the same for both these techniques. Besides, the same base learner is used for WCE and ActMiner. The parameter settings for OLINDDA are the same as in [4].

In this experiment, we also use WCE-OLINDDA_Parallel and WCE-OLINDDA_Single for comparison, with some minor changes. In order to see the effects of limited labeled data on WCE-OLINDDA models, we run two different settings for WCE-OLINDDA_Parallel and WCE-OLINDDA_Single. First, we run WCE-OLINDDA_Parallel (WCE-OLINDDA_Single) with *all instances in each chunk labeled*. We denote this setting as WCE-OLINDDA_Parallel-Full (WCE-OLINDDA_Single-Full). Second, we run WCE-OLINDDA_Parallel

(WCE-OLINDDA_Single) with exactly the same instances labeled as were labeled by ActMiner, *plus* any instance identified as novel class by WCE-OLINDDA_Parallel (WCE-OLINDDA_Single). We denote this setting as WCE-OLINDDA_Parallel-Partial (WCE-OLINDDA_Single-Partial). We will henceforth use the acronyms **AM** for ActMiner, $\mathbf{WOP}_f$ for WCE-OLINDDA_Parallel-Full, $\mathbf{WOS}_f$ for WCE-OLINDDA_Single-Full, $\mathbf{WOP}_p$ for WCE-OLINDDA_Parallel-Partial, and $\mathbf{WOS}_p$ for WCE-OLINDDA_Single-Partial.

4.3 Evaluation

Evaluation approach: Let F_n = total novel class instances misclassified as existing class, F_p = total existing class instances misclassified as novel class, F_e = total existing class instances misclassified (other than F_p), N_c = total novel class instances in the stream, N = total instances the stream. We use the following performance metrics to evaluate our technique: $\boldsymbol{M_{new}}$ = % of novel class instances Misclassified as existing class = $\frac{F_n*100}{N_c}$, $\boldsymbol{F_{new}}$ = % of existing class instances Falsely identified as novel class = $\frac{F_p*100}{N-N_c}$, **ERR** = Total misclassification error (%)(including M_{new} and F_{new}) = $\frac{(F_p+F_n+F_e)*100}{N}$. From the definition of the error metrics, it is clear that ERR is not necessarily equal to the sum of M_{new} and F_{new}. Also, let L_p be the percentage of instances in the data stream required to have labels for training.

Evaluation is done as follows: we build the initial models in each method with the first *init_number* labeled chunks with all instances in each chunk labeled. In our experiments, we set *init_number* = 3. From the 4th chunk onward, we evaluate the performances of each method on each data point. We update the models with a new chunk whenever all weakly classified instances (WCIs) in that chunk are labeled.

Results: Figures 1(a_1),1(b_1) show the ERR of each baseline technique and figures 1(a_2),1(b_2) show the percentage of data labeled (L_p) corresponding to each technique on a real (Forest) and a synthetic (SynCN) dataset with decision tree. Corresponding charts for other datasets and k-NN classifier are similar, and omitted due to the space limitation. Figure 1(a_1) shows the ERR of each technique at different stream positions for Forest dataset. The X axis in this chart corresponds to a particular stream position, and the corresponding value at the Y axis represents the ERR upto that position. For example, at X=200, corresponding Y values represent the ERR of a technique on the first 200K instances in the stream. At this position, corresponding Y values (i.e., ERR) of AM, WOP$_f$, WOP$_P$, WOS$_f$ and WOS$_p$ are 7.5%, 10.8%, 56.2%, 12.3%, and 63.2%, respectively. The percentage of data required to be labeled (L_P) by each of these techniques for the same dataset (Forest) is shown in figure 1(a_2). For example, at the same X position (X=200), the L_P values for AM, WOP$_f$, WOP$_P$, WOS$_f$ and WOS$_p$ are 8.9%, 100%, 12.7%, 100%, and 9%, respectively. Therefore, from the first 200K instances in the stream, AM required only 8.9% instances to have labels, whereas, its nearest competitor (WOP$_f$) required 100% instances to

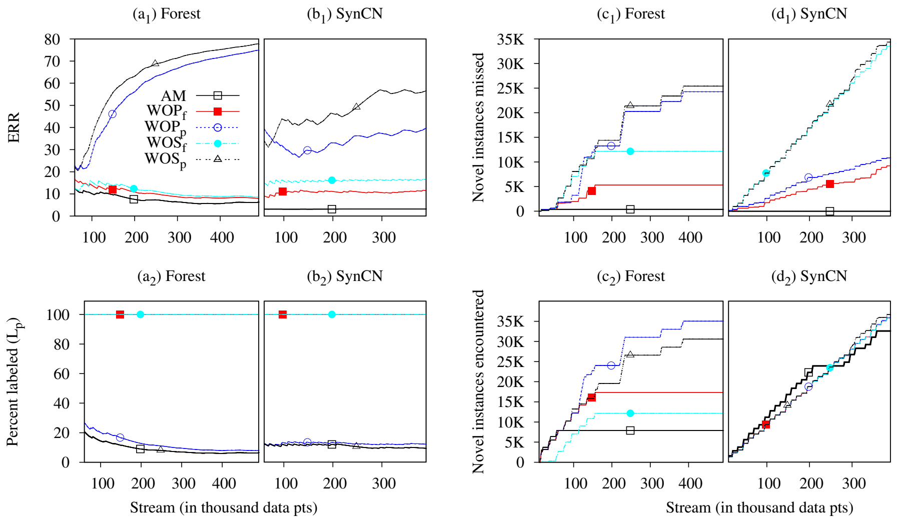


Fig. 1. Overall error (ERR), percentage of data required to be labeled (L_p), total novel instances missed, and encountered by each method

have labels. So, AM, using 11 times less labeled data, achieves lower ERR rates than WOP$_f$. Note that ERR rates of other methods such as WOP$_p$, which uses less than 100% labeled data, are much worse.

Figures 1(c$_1$),1(d$_1$) show the number of novel instances missed (i.e., misclassified as existing class) by each baseline technique, and figures 1(c$_2$),1(d$_2$) report the total number of novel instances encountered by each technique on the same real (Forest) and synthetic (SynCN) datasets with decision tree classifier. For example, in figure 1(c$_1$), for X=200, the Y values represent the total number of novel class instances missed by each technique within the first 200K instances in the stream. The corresponding Y values for AM, WOP$_f$, WOP$_P$, WOS$_f$ and WOS$_p$ are 366, 5,317, 13,269, 12,156 and 14,407, respectively. figure 1(c$_2$) shows the total number of novel instances encountered by each method at different stream positions for the same dataset. Different approaches encounter different amount of novel class instances because the ensemble of classifiers in each approach evolve in different ways. Therefore, a class may be novel for one approach, and may be existing for another approach.

Table 1 shows the summary of the evaluation. The table is split into two parts: the upper part shows the ERR and M_{new} values, and the lower part shows the F_{new} and L_p values. For example, consider the upper part of the table corresponding to the row *KDD* under *Decision tree*. This row shows the ERR and M_{new} rates for each of the baseline techniques on KDD dataset for decision tree classifier. Here AM has the lowest ERR rate, which is 1.2%, compared to 5.8%, 64.0%, 6.7%, and 74.8% ERR rates of WOP$_f$, WOP$_p$, WOS$_f$ and WOS$_p$, respectively. Also, the M_{new} rate of AM is much lower (1.4%) compared to any other baselines. Although WOS$_f$ and WOP$_f$ have lower ERR rates in SynC

Table 1. Performance comparison

Classifier	Dataset	ERR					M_{new}				
		AM	WOP$_f$	WOP$_p$	WOS$_f$	WOS$_p$	AM	WOP$_f$	WOP$_p$	WOS$_f$	WOS$_p$
Decision tree	SynC	13.4	14.1	42.5	**12.8**	42.3	-	-	-	-	-
	SynCN	**0.3**	8.9	38.4	13.9	55.7	**0.0**	26.5	31.0	96.2	96.3
	KDD	**1.2**	5.8	64.0	6.7	74.8	**1.4**	13.2	22.4	96.9	96.1
	Forest	**6.3**	7.9	74.5	8.5	77.4	**4.6**	30.7	69.3	70.1	83.1
k-NN	SynC	**0.0**	2.4	2.4	1.1	1.1	-	-	-	-	-
	SynCN	**0.0**	8.9	17.2	13.9	36.0	**0.0**	26.5	26.3	96.2	98.9
	KDD	**1.1**	4.9	15.3	5.2	63.2	**6.2**	12.9	76.1	96.5	99.1
	Forest	7.1	**4.1**	16.9	4.6	37.8	**15.4**	32.0	28.6	70.1	82.2
Classifier	Dataset	F_{new}					L_p				
		AM	WOP$_f$	WOP$_p$	WOS$_f$	WOS$_p$	AM	WOP$_f$	WOP$_p$	WOS$_f$	WOS$_p$
Decision tree	SynC	**0.0**	2.4	2.4	1.1	1.0	**1.04**	100	3.41	100	2.05
	SynCN	**0.0**	1.6	1.5	0.1	0.1	**9.31**	100	12.10	100	9.31
	KDD	1.1	4.3	4.5	**0.03**	0.03	**3.33**	100	8.82	100	3.34
	Forest	3.0	1.1	1.1	**0.2**	0.2	**6.51**	100	8.08	100	6.56
k-NN	SynC	**0.0**	2.4	2.4	1.1	1.1	**0.0**	100	2.46	100	1.09
	SynCN	**0.0**	1.6	1.7	0.1	0.1	**8.35**	100	12.73	100	8.35
	KDD	0.9	4.4	4.8	**0.03**	0.03	**1.73**	100	7.94	100	1.73
	Forest	1.9	1.1	1.0	**0.2**	0.2	**5.05**	100	6.82	100	5.20

(decision tree), and Forest (k-NN), respectively, they use at least 20 times more labeled data than AM in those datasets, which is reported in the lower right part of the table (under L_p), and their M_{new} rates are much higher than AM. Note that L_p is determined from the WCIs, i.e., what percentage of instances are weakly classified by the ensemble. Therefore, it is different for different datasets. Some readers might find it surprising that active learning outperforms learning with full labels. However, it should be noted that ActMiner outperforms other proposed techniques, not MineClass itself. MineClass, using 100% labeled instances for training, still outperforms ActMiner because ActMiner uses less labeled instance. However, other proposed techniques (like WOP) have too strong requirement about class properties. For example, OLINDDA requires the classes to have convex shape, and assumes similar density of each class of data. On the other hand, ActMiner does not have any such requirement. Therefore, in most real world scenarios, where classes have non-convex shape, and different classes have different data densities, ActMiner performs much better than OLINDDA in detecting novel class, even with much less label information.

Figure 2(left) shows the effect of increasing the minimum majority threshold (MMT) on ERR rate, and figure 2(right) shows the percentage instances labeled for different values of MMT on SynC. For AM, the ERR rate starts decreasing after MMT=0.5. This is because there is no instance for which the M2S (majority to sum) ratio is less than 0.5. So, L_p remains the same (1%) for MMT=0.1 to 0.5 (see figure 2(b)), since the only instances needed to be labeled for these values of MMT are the *F-outlier* instances. However, when MMT=0.6, more instances needed to be labeled (L_p=3.2%) as the M2S ratio for these (3.2-1.0=) 2.2% instances are within the range [0.5,0.6). The overall ERR also reduces since more labeled instances are used for training. Sensitivity of AM to other parameters are similar to MineClass [4], and omitted here due to space limitations.

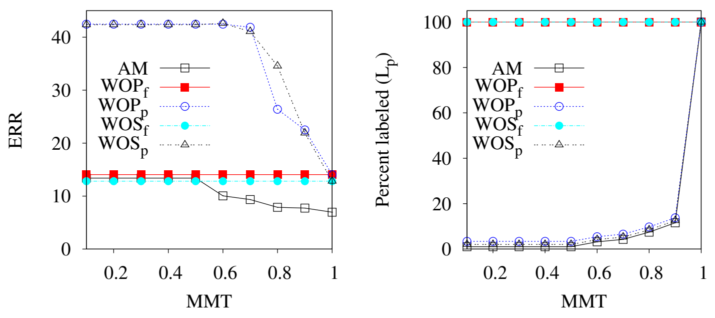


Fig. 2. Effects of increasing the minimum majority threshold (MMT)

Table 2. Running time comparison in all datasets

Dataset	Time(sec)/1K			Time(sec)/1K (including labeling time)		
	AM	WOP$_f$	WOS$_f$	AM	WOP$_f$	WOS$_f$
SynC	0.32	0.41	**0.2**			
SynCN	**1.6**	14.3	3.1			
KDD	1.1	24.0	**0.6**	**34.4**	1,024.0	1,000.6
Forest	0.87	8.5	**0.5**	**66.0**	1,008.5	1,000.5

Table 2 reports the running times of AM and other baseline techniques on different datasets with decision tree. Running times with k-NN also have similar characteristics. Since WOP$_f$ and WOP$_p$ have the same running times, we report only WOP$_f$. The same is true for WOS$_f$ and WOS$_p$. The columns headed by "Time (sec)/1K " show the average running times (train and test) in seconds per 1000 points excluding data labeling time, and the columns headed by "Time (sec)/1K (including labeling time)" show the same including data labeling time. For example, excluding the data labeling time, AM takes 1.1 seconds to process 1K instances on the KDD dataset, whereas WOP$_f$, WOP$_p$ takes 24.0, and 0.5 seconds, respectively. In general, WOP$_f$ is much slower than AM, requiring about C times more runtime than AM. This is because WOP maintains C parallel $OLINDDA$ models to detect novel classes. Besides, OLINDDA creates clusters using an internal buffer every time it encounters an instance that is identified as *unknown*, which consumes much of its running time. On the other hand, WOS$_f$ runs slightly faster than AM in three datasets. But this advantage of WOS$_f$ is undermined by its much poorer performance in classification accuracy than AM. If we consider the data labeling time, we get a more compelling picture. We consider the labeling times only for real datasets. Suppose the labeling time for each data point for the real datasets is 1 sec, although in real life, data labeling may require much longer time [12]. Out of each 1000 instances, AM requires only 33, and 65 instances to have labels for the KDD, and Forest datasets, respectively (see table 1 under L_p). Whereas WOP$_f$ and WOS$_f$ require all the 1000 instances to have labels. Therefore, the total running time of AM per 1000 instances including data labeling time is only 3.4% and 6.5% of that of WOP_f and WOS_f for KDD and Forest datasets, respectively. Thus, AM outperforms the baseline techniques both in classification accuracies and running times.

5 Conclusion

Our approach, ActMiner, provides a more complete framework for data stream classification than existing techniques. ActMiner integrates the solutions to four major data stream classification problems: infinite length, concept-drift, concept-evolution, and limited labeled data. Most of the existing techniques address only two or three of these four problems. ActMiner reduces data labeling time and cost by requiring only a few selected instances to be labeled. Even with this limited amount of labeled data, it outperforms state-of-the-art data stream classification techniques that use ten times or more labeled data. In future, we would like to address the dynamic feature set problem and multi-label classification problems in data stream classification.

References

1. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proc. KDD '01, pp. 97–106 (2001)
2. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proc. KDD '03, pp. 226–235 (2003)
3. Chen, S., Wang, H., Zhou, S., Yu: Stop chasing trends: Discovering high order models in evolving data. In: Proc. ICDE '08, pp. 923–932 (2008)
4. Masud, M.M., Gao, J., Khan, L., Han, J., Thuraisingham, B.M.: Integrating novel class detection with classification for concept-drifting data streams. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009. LNCS, vol. 5782, pp. 79–94. Springer, Heidelberg (2009)
5. Kolter, J.Z., Maloof, M.A.: Using additive expert ensembles to cope with concept drift. In: Proc. ICML '05, Bonn, Germany, pp. 449–456 (2005)
6. Yang, Y., Wu, X., Zhu, X.: Combining proactive and reactive predictions for data streams. In: Proc. KDD '05, pp. 710–715 (2005)
7. Zhu, X., Zhang, P., Lin, X., Shi, Y.: Active learning from data streams. In: Perner, P. (ed.) ICDM 2007. LNCS (LNAI), vol. 4597, pp. 757–762. Springer, Heidelberg (2007)
8. Fan, W., an Huang, Y., Wang, H., Yu, P.S.: Active mining of data streams. In: SDM, pp. 457–461 (2004)
9. Masud, M.M., Gao, J., Khan, L., Han, J., Thuraisingham, B.M.: A practical approach to classify evolving data streams: Training with limited amount of labeled data. In: Proc. ICDM '08, pp. 929–934 (2008)
10. Spinosa, E.J., de Leon, F., de Carvalho, A.P., Gama, J.: Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks. In: Proc. SAC '08, pp. 976–980 (2008)
11. Tumer, K., Ghosh, J.: Error correlation and error reduction in ensemble classifiers. Connection Science 8(304), 385–403 (1996)
12. van Huyssteen, G.B., Puttkammer, M.J., Pilon, S., Groenewald, H.J.: Using machine learning to annotate data for nlp tasks semi-automatically. In: Proc. Computer-Aided Language Processing, CALP'07 (2007)

Bulk Loading Hierarchical Mixture Models for Efficient Stream Classification

Philipp Kranen, Ralph Krieger, Stefan Denker, and Thomas Seidl

Data Manangement and Exploration Department
RWTH Aachen University, Germany
{kranen,krieger,denker,seidl}@cs.rwth-aachen.de

Abstract. The ever growing presence of data streams led to a large number of proposed algorithms for stream data analysis and especially stream classification over the last years. Anytime algorithms can deliver a result after any point in time and are therefore the natural choice for data streams with varying time allowances between two items. Recently it has been shown that anytime classifiers outperform traditional approaches also on constant streams. Therefore, increasing the anytime classification accuracy yields better performance on both varying and constant data streams. In this paper we propose three novel approaches that improve anytime Bayesian classification by bulk loading hierarchical mixture models. In experimental evaluation against four existing techniques we show that our best approach outperforms all competitors and yields significant improvement over previous results in term of anytime classification accuracy.

1 Introduction

With an abundance of streaming data due to widely deployed sensors or other data gathering devices, analysis of streaming data such as stream classification has recently received much attention in data mining research. Algorithms that work on data streams have to cope with the limited and often also varying amount of computation time. Traditionally they got for a certain task a fixed time budget which was known in advance, i.e. they were tailored to the specific application. These budget algorithms can neither provide a results in less time nor exploit additional time to improve their result. In contrast, so called anytime algorithms can provide a result after a very short initialization, improve their result incrementally when more time is available and hold the most recent result ready at any time. In data mining anytime solutions have been proposed for many tasks such as clustering [10], top-k processing [2] and classification [5,16].

Anytime algorithms are the natural choice for varying data streams since they flexibly exploit all available time to improve the quality of their result. Recently it has been shown in [12] that also on constant data streams anytime classifiers can improve the classification accuracy over that of traditional budget approaches. With their superiority on varying and constant data streams, applications for anytime classifiers are numerous and range from industrial applications, such as machine monitoring, over sorting tasks to robotics and health applications. [11]. Our focus is on improving the performance of anytime Bayesian classification. Improving the anytime accuracy together with the results from [12] leads to better classification results on both constant and varying data streams.

2 Related Work

Classification aims at determining the class label of unknown objects based on training data. Different classification approaches are discussed in the literature including nearest neighbor classifiers, decision trees or support vector machines. Bayes classifiers constitute a statistical approach that has been successfully used in numerous application domains. Another classification approach is represented by Bayesian classification using kernel density estimation [9]. Especially for huge data sets the estimation error using kernel densities is known to be very low and even asymptotically optimal [3].

Anytime classification is real time classification up to a point of interruption. In addition to high classification accuracy as in traditional classifiers, anytime classifiers have to make best use of the limited time available, and, most notably, they have to be interruptible at any given point in time. This point in time is usually not known in advance and may vary greatly. Anytime classification has for example been discussed for support vector machines [5] or nearest neighbor classification [16].

For Bayesian classification based on kernel densities an anytime algorithm called Bayes tree has been proposed in [15]. The Bayes tree is a balanced tree structure and is basically an extension of the R-tree [8]. It stores in each entry a pointer and a minimum bounding rectangle (MBR) and additionally a cluster feature representing the corresponding subtree. In [15] the tree is constructed trough iterative insertion, i.e. no optimization is performed with respect to overlapping or quality of the resulting mixture densities. In this paper we propose several methods for bulk loading mixture densities in the Bayes tree and show in experimental evaluation that they outperform the results from [15].

3 Bulk Loading Mixture Densities

Before going into detail on the different bulk loading approaches in Sections 3.2 and 3.3 we briefly review Bayesian classification and describe the structure and working of the Bayes tree proposed in [15].

3.1 Bayesian Classification and the Bayes Tree

Given a set of classes C and an object space X a classifier is a function G that assigns the class label $G(x)$ to an object $x \in X$. Based on a statistical model of the distribution of class labels the Bayes classifier assigns to an object x the class c_i with the highest posterior probability $P(c_i|x)$. With Bayes rule it holds:

$$G(x) = \underset{c_i \in C}{argmax}\{P(c_i|x)\} = \underset{c_i \in C}{argmax}\{P(c_i) \cdot p(x|c_i)\}$$

In the Bayes tree the class-conditional density $p(x|c_i)$ is estimated using Gaussian mixture models in the inner nodes and Gaussian kernel estimators at the leaf level. Iterative refinement of mixture components enables anytime kernel density estimation for efficient and interruptible classification. The general idea of the Bayes tree is a hierarchy of mixture densities stored in a multidimensional index. Each level of the tree stores a complete model of the entire data at a different granularity. The node entries consist of pointers to a subtree and a single Gaussian representing the objects in the subtree. All objects stored in the leaves of the Bayes tree are d-dimensional kernels. The mean μ_s and the variance vector σ_s^2 can be computed from the cluster features.

Answering a probability density query uses a complete model which is available at each level of the tree. Besides these full models, the Bayes tree allows for local refinement of the model (to adapt flexibly to the query) and thus provides models composed of coarser and finer representations. The current mixture model components, i.e. their corresponding entries, are stored in a *frontier.* The current entries in the frontier have to represent each stored object exactly once. This is made sure by removing the entry e_s that is refined from the frontier and adding its child entries $e_{s \circ j}, j = 1 \ldots \nu_s$ instead. The probability density for a query object is then calculated with respect to the current frontier.

For tree traversal best first descent using a probabilistic priority measure has proven to yield the best results in [15]. One Bayes tree is built per class, therefore several improvement strategies have been proposed to decide which tree has the right to refine its model in the next time step. Extensive experiments showed that refining the k most probable classes (qbk) in turns yielded the best results throughout. $k = \min\{2, \lfloor \log(m) \rfloor\}$, where m is the number of classes, showed the best performance on all tested data sets. For more details please refer to [15].

3.2 Machine Learning and Statistical Approaches

Our goal in this work is to improve the performance of the Bayes tree. The accuracy of the Bayes tree results is based on the quality of the mixture densities stored in its entries. The iterative insertion performed in [15] does not consider the quality of the resulting Gaussian components. We develop and evaluate several bulk loading approaches that try to overcome this shortcoming and improve the quality of the mixture densities.

Goldberger. Since the Bayes tree is a statistical approach to classification we looked for statistical methods to create a smaller mixture model from a given mixture model. Starting bottom up with a mixture model that contains a kernel estimator for each training set item we create successively coarser models that represent good approximations.

Our first statistical approach is based on [7] and is called Goldberger in the following. The Goldberger approach assumes two initial mixture models f and g to be given, where f is the finer model with r components and g an approximation with s components, hence $r > s$. Each component is assigned a weight and is specified by its mean and covariance matrix. To measure the quality of the approximation [7] defines the distance between two mixture densities as:

Definition 1. *Let $f = \sum_{i=1}^{r} \alpha_i f_i$ and $g = \sum_{j=1}^{s} \beta_j g_j$ be two mixture densities containing r and s Gaussian components f_i and g_j with their respective weights α_i and β_j. The distance between f and g is then defined using the Kullback-Leibler divergence KL [4] as follows*

$$d(f,g) = \sum_{i=1}^{r} \alpha_i \cdot \min_{j=1}^{s}\{KL(f_i, g_j)\}$$

The optimal model $\hat{g}$ reducing f to s components is $\hat{g} = \arg\min_g(d(f,g))$. Since there is no closed form to compute $\hat{g}$, a local optimum is computed iterating the following two steps until the distance $d(f,g)$ does no longer decrease. Therein $\pi(i) : \{1 \dots r\} \to \{1 \dots s\}$ is a mapping function that assigns each component in f to a component in g.

– Regroup: update π: $\pi(i) = \arg\min_{j=1}^{s}\{KL(f_i, g_j)\}$
– Refit: for each component g_j recompute weight β_j, mean μ_j and covariance matrix Σ_j as follows
  - $\beta_j = \sum_{i,\pi(i)=j} \alpha_i$
  - $\mu_j = \frac{1}{\beta_j} \sum_{i,\pi(i)=j} \alpha_i \mu_i$
  - $\Sigma_j = \frac{1}{\beta_j} \sum_{i,\pi(i)=j} \alpha_i \left(\Sigma_j + (\mu_i - \mu_j)^2\right)$

We devise a bulk loading technique based on [7] as follows. To initialize the mixture g we compute a first mapping π_0 by assigning $0.75 \cdot M$ components from f to one component in g according to the z-curve order of their mean values. M is given through the fanout, which in turn is dictated by the page size. When no more changes occur in step 2, the resulting components g_j are converted to Bayes tree nodes containing the entries f_i with $\pi(i) = j$. Since the final π might map more than M components from f to a single component in g, we investigated several strategies to restrict the fanout to the given boundaries. First we reformulated the regroup step into an integer linear program with constraints regarding the resulting fanout. However, for realistic problem sizes, this approach took way too long to compute a complete bulk loading. Hence, we decided for a post processing after the mapping π was computed, which splits the nodes that contain too many entries. Therefore two representatives are computed by moving the mean along the dimension a with the highest variance σ_a by an $\epsilon = \sigma_a/2$ in both direction. A Gaussian is placed over the two representatives and the mapping of the entries to the representatives is computed as in the regroup step. If a node contains too few entries it is merged with the node closest to it in terms of the Kullback-Leibler divergence.

Virtual sampling. The second approach, called virtual sampling, uses the work presented in [17] and does not rely on the KL divergence. The virtual sampling approach assumes a given mixture model $f = \sum_{i=1..r} \alpha_i \cdot f_i$ containing r components and computes a coarser mixture model $g = \sum_{j=1..s} \alpha_j \cdot g_j$ with $s < r$ components. The components $f_i = \mathcal{G}(x, \mu_i, \sigma_i)$ constitute multivariate Gaussian normal distributions with their respective weight α_i (analogue for g_j). To derive an algorithm the following model is utilized: the mixture g can be computed using samples $R_1 \dots R_r$ from each component in f with $R = \cup_{i=1..r} R_i$ and $|R_i| = \alpha_i \cdot |R|$. Assuming independence of the sample points from different components in f yields the assumption that they can be assigned to different components in g while samples from the same f_i are likely to be assigned to the same g_j. Based on this assumption hidden variables z_{ij} are introduced that indicate for each component f_i its assignment to the corresponding g_j. While the z_{ij} are binary during initialization, they can take values between 0 and 1 during the iterations. The hidden variables are used in a modified Expectation Maximization algorithm to compute the coarser mixture g as follows (superscripts ^{f} and ^{g} are added for readability to indicate the origin of the components):

- Expectation: $z_{ij} = \frac{\left[\mathcal{G}(\mu_i^f, \mu_j^g, \Sigma_j^g) e^{-\frac{1}{2} trace\{(\Sigma_j^g)^{-1} \Sigma_i^f\}}\right]^{|R_i|} \cdot \alpha_j^g}{\sum_{k=1}^{s} \left[\mathcal{G}(\mu_i^f, \mu_k^g, \Sigma_k^g) e^{-\frac{1}{2} trace\{(\Sigma_k^g)^{-1} \Sigma_i^f\}}\right]^{|R_i|} \cdot \alpha_k^g}$
- Maximization:
  - $\alpha_j^g = \frac{1}{r} \sum_{i=1}^{r} z_{ij} \qquad \mu_j^g = \frac{\sum_{i=1}^{r} z_{ij} |R_i| \mu_i^f}{\sum_{i=1}^{r} z_{ij} |R_i|}$
  - $\Sigma_j^g = \frac{1}{\sum_{i=1}^{r} z_{ij} |R_i|} \left[\sum_{i=1}^{r} z_{ij} |R_i| \Sigma_i^f + \sum_{i=1}^{r} z_{ij} |R_i| \left(\mu_i^f - \mu_j^g\right)^2\right]$

The above equations are independent of the actual samples R_i and can be computed directly from the mixture components in f, hence *virtual sampling*. To use the described bottom up method for bulk loading we have to provide an initialization for the hidden variables z_{ij}. The initialization of the mixture g is done as in the goldberger approach described above. After getting the final values for z_{ij} from the virtual sampling algorithm, we assign each f_i to that g_j with the maximum z_{ij} for all j. Moreover, the result has to comply with the fanout parameters m and M of the Bayes tree. This is achieved through merging and splitting of the resulting components g_j. If a component g_j is assigned less than m components f_i, these components from f are assigned to the $g_{j'}$ with the second highest $z_{ij'}$. If more than M components f_i are assigned to one g_j, g_j is duplicated while moving the resulting two means in opposite direction along the dimension with the highest variance. The respective f_i are reassigned to the more probable candidate according to the density of their mean μ_i. After merging and splitting the corresponding mixture parameters are adapted following the above equations.

EMTopDown. Besides the above mentioned bottom up approaches we implemented a top down approach that recursively splits the training set into several clusters. In contrast to the previous approach, where Gaussian components were merged and mapped, we now operate solely on the data objects. More precisely, we start by applying the EM [6] algorithm to the complete training set. The desired number M of resulting clusters is always set to the fanout which is again given through the page size. If the EM returns less than m clusters, the biggest resulting cluster is split again such that the total number of resulting clusters is at most M. In the rare case that the EM returns a single cluster, this cluster is split by picking the two farthest elements and assigning the remaining elements to the closest of the two. Finally, if a resulting cluster contains more than L objects (the capacity of a leaf node), the cluster is recursively split using the procedure described above. Otherwise the items contained in that cluster are stored in a leaf node, its corresponding entry is calculated and returned to build the Bayes tree. The EM approach may result in an unbalanced tree, which differs from the primary Bayes tree idea. However, as we will see in the experimental section, the results show that this is not a drawback but even leads to better anytime classification performance.

3.3 Data Base Driven Approaches

Since the Bayes tree extends the R-tree, we employ traditional R-tree bulk loading algorithms for comparison. We implemented two types of space filling curves, namely Hilbert curve and z-curve. We briefly describe the Hilbert curve approach, the z-curve bulk loading works analogously. The bulk loading according to the Hilbert curve is a bottom up approach where in the first step the Hilbert value for each training set item is calculated. Next the items are ordered according to their Hilbert value and put into leaf nodes w.r.t. the page size. After that the corresponding entry for each resulting node is created, i.e. MBR, cluster features (CF) and the pointer. These steps are repeated using the mean vectors as representatives until all entries fit into one node, the root node. Theory on creating multidimensional Hilbert curves can be found in [1], for implementation guide lines see [13]. Additionally we implemented the partitioning approach presented in [14] that is called sort-tile-recursive. The basic idea is to build a hierarchy of rectangles which have, at the same level of the hierarchy, approximately the same expansion in each dimension. For details please refer to [14].

4 Experiments

The three proposed bulk loading techniques *Goldberger*, *virtual sampling* and *EMTopDown* are compared to the existing R-tree bulk loading approaches *Hilbert*, *z-curve* and *STR* and the previous results from [15] (called *Iterative* in the graphs since it performs iterative insertion of objects). We also used the same settings as in [15], i.e. we use the same data sets, performed 4-fold cross validation and show the classification accuracy after each node averaged over the four folds. We used global best descent and the *qbk* improvement strategy as they showed the best results in [15]. Please note that the bulk loading is done per fold once and offline and the resulting classifier is then used on the data stream. Since our focus is on anytime classification, we do not study the time performance of the bulk loading algorithm but the performance of the resulting classifier, i.e. its anytime classification accuracy.

The top left part of figure 1 shows the results for the pendigits data set. The Goldberger approach fails to improve the accuracy over the iterative insertion for the first 50 nodes. After that it performs slightly better, but cannot increase the accuracy by more than 1%. Virtual sampling performs worst on this data set.

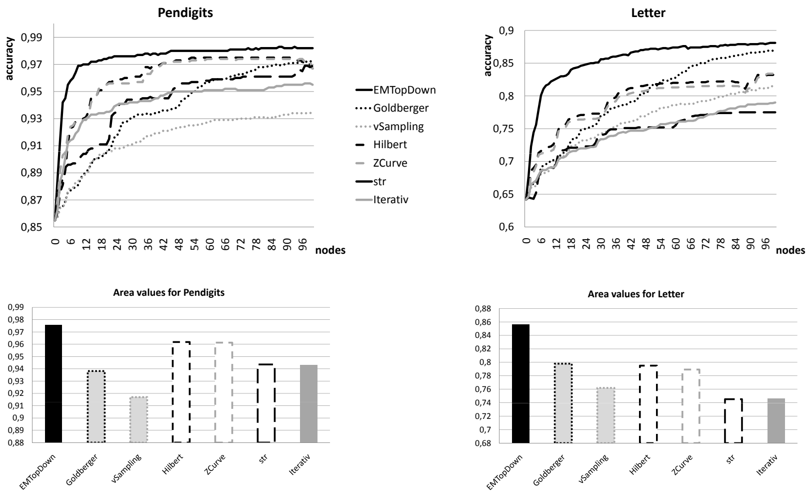


Fig. 1. Anytime classification accuracy and a ranking of all approaches according to the area for Pendigits (left) and Letter (right)

The Hilbert and z-curve bulkload yield comparable results, their corresponding curves show a steep increase similar to the iterative insertion and show better performance in most cases. After falling behind during the first nodes, STR performs equaly well compared to Iterative. The EMTopDown bulkload outperforms all other approaches and improves the accuracy over the iterative insertion constantly by 3% or more on this data set.

The performance of the Goldberger bulkload stayed below the iterative insertion in the majority of our experiments. Just on the Letter data set it improved the accuracy for larger time allowances (cf. Figure 1, right). For the first 40 nodes Goldberger and Iterative perform equally well, after that the accuracy of Iterative stays behind that of Goldberger. While the virtual sampling and STR bulkload shows similar performance to Iterative, Hilbert and z-curve (which are again in close proximity to each other) show constantly better accuracy than Iterative. The EMTopDown again constantly yields the best accuracy up to 13% better than the iterative insertion.

To facilitate an easier comparison between the different approaches we report the values for the normalized area under the anytime curves for Pendigits, Letter, Vowel, USPS and Verbmobil in Figures 1 (bottom) and 2 (top) respectively. Throughout the data sets Hilbert and z-curve show nearly the same performance, while z-curve is usually slightly behind Hilbert except for the USPS data set. STR ranges between these two and the iterative insertion; it is never better than the former and never beaten by the latter. Surprisingly both statistical approaches exhibit the same weakness as STR, i.e. they never outperform the z-curve bulk load (except Goldberger on Letter) and several times show even worse performance than iterative insertion. This is especially interesting since both approaches are initialized using the z-curve, however, only with $0.75 \cdot M$ entries per node (cf. Section 3.2). We discuss the reasons for this shortcoming of

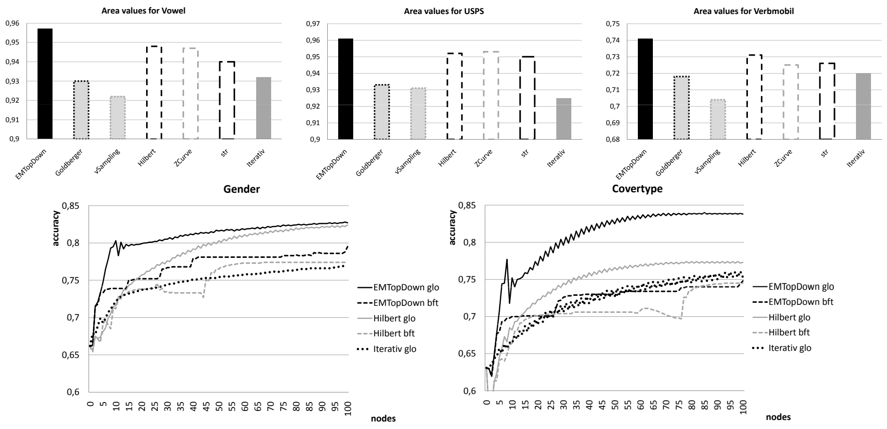


Fig. 2. Top: Comparison of all approaches on Vowel, USPS and Verbmobil data sets. Bottom: Anytime classification accuracy on Gender (left) and Covertype (right).

the statistical approaches at the end of the section where we analyze the structure of the resulting trees. Finally, the EMTopDown bulk load shows constantly the best performance on all data sets despite the unbalanced resulting trees. Again we defer the analysis to the end of the section and first discuss a different issue.

Figure 2 (bottom) shows the results for the gender and covertype data sets. For readability only the results for Hilbert and EMTopDown are shown. For both data sets $k = 2$ for the qbk improvement strategy (cf. Section 3.1). The graphs for EMTopDown and Hilbert using the global best descent (*glo*) show an oscillating behavior on both data sets. For comparison we recapitulated the breadth first traversal (*bft*), the results are plotted as well. As was found in [15], the global best descent performs better than breadth first traversal. However, the graphs for bft do not show the oscillating behavior mentioned above. Since $k = 2$, there is obviously a certain percentage of object whose class decision changes in favor of (or against) the tree which is currently refined. More precisely, these objects are likely positioned on the decision boundary between the two most probable classes. In global best descend refining mixture components close to the objects, and hence close to the decision boundary, affects the corresponding posterior probabilities more heavily than refinement of a farther component as in breadth first traversal. If we assume the oscillation to be a higher frequency added to a smooth underlying anytime curve, the percentage of these borderline objects corresponds to the amplitude. However, on balance, the oscillating behavior does not affect the superiority of the bulk loading over the iterative insertion.

To find reasons for the surprising ranking of the individual algorithms, we analyzed the structure of the resulting trees. Since more entries in a node correspond to more detailed information compared to less entries, we looked at the degree to which the nodes were filled in the different approaches. To this end we computed the average fanout per level of the trees from root to leaf. However, the resulting figures did not reveal any correlation to the found ranking of the approaches. For example, both space filling curve approaches always fill the nodes

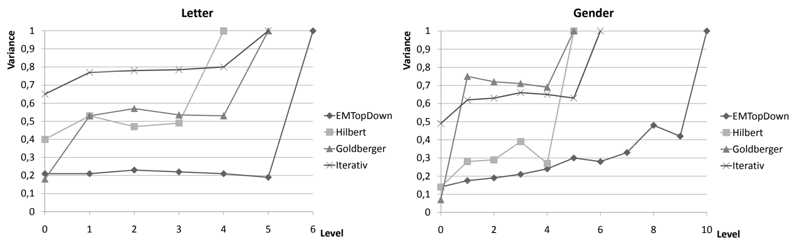


Fig. 3. Variances per level for Letter (left) and Gender (right)

to nearly 100% (except for the last one per level and the root), but the EMTopDown sometimes produces less than M entries for a given node.

We found a correlation between the performance of the algorithm and the variance of the mixture components in the resulting trees. More precisely, we calculated the average variance of all entries per level, Figure 3 shows the resulting numbers for Hilbert, Goldberger, EMTopDown and Iterative on the Letter and Gender data sets. The variances are normalized by the variance of the entire data set per class. Level 0 corresponds to the leaves, Level 1 is above the leaves etc. The trees resulting from different approaches can have different heights as can be seen in the graphs. Hilbert bulk load fills each node to 100% and consequently yields the smallest trees, while the unbalanced trees resulting from EMTopDown are up to twice as high on the Gender data set.

EMTopDown and Hilbert show significantly smaller average variances compared to the iterative insertion. While the corresponding variances for Goldberger are smaller than those of Iterative for the Letter data set they are larger on the Gender data set. This is in line with the observed anytime classification performances. We found comparable similarities between the two measures on the other data sets. The average variances per level achieved by the EMTopDown bulk load were constantly amongst the lowest compared to all other approaches. This explains and underlines the superior performance of EMTopDown.

In general the EMTopDown shows the best results in terms of anytime classification accuracy on all tested data sets and continuously improves the accuracy over that of the previous results in [15] up to 13%. This proves the effectiveness of our bulk loading approach for hierarchical anytime classifiers.

5 Conclusion

We proposed three bulk loading approaches for hierarchical mixture models to improve Bayesian classification on data streams using the Bayes tree. We compared our approaches to the previously proposed iterative insertion [15] and three known R-tree bulk loading algorithms on a range of real world data sets. Experimental results showed that our novel EMTopDown bulk load constantly outperformed all other approaches and improved the accuracy by up to 13%. Surprisingly our two statistical approaches were outperformed by existing R-tree bulk loadings based on space filling curves. Further analysis attributed this shortcoming to a structural property of the resulting Bayes trees. The results of the analysis were in line with the classification results found in the experiments confirming the superior performance of our new EMTopDown bulk loading in terms of anytime classification accuracy.

Acknowledgments. This work has been supported by the UMIC Research Centre, RWTH Aachen University.

References

1. Alber, J., Niedermeier, R.: On multi-dimensional hilbert indexings. In: 4th Annual International Conference on Computing and Combinatorics COCOON (1998)
2. Arai, B., Das, G., Gunopulos, D., Koudas, N.: Anytime measures for top-k algorithms on exact and fuzzy data sets. VLDB Journal 18(2), 407–427 (2009)
3. Bouckaert, R.: Naive Bayes Classifiers that Perform Well with Continuous Variables. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 1089–1094. Springer, Heidelberg (2004)
4. Chen, J.-Y., Hershey, J., Olsen, P., Yashchin, E.: Accelerated monte carlo for kullback-leibler divergence between gaussian mixture models. In: ICASSP (2008)
5. DeCoste, D.: Anytime interval-valued outputs for kernel machines: Fast support vector machine classification via distance geometry. In: ICML (2002)
6. Dempster, A.P., Laird, N.M.L., 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. Goldberger, J., Roweis, S.T.: Hierarchical clustering of a mixture model. In: NIPS (2004)
8. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: SIGMOD, pp. 47–57 (1984)
9. John, G., Langley, P.: Estimating continuous distributions in bayesian classifiers. In: UAI. Morgan Kaufmann, San Francisco (1995)
10. Kranen, P., Assent, I., Baldauf, C., Seidl, T.: Self-adaptive anytime stream clustering. In: Proc. of the 9th IEEE ICDM (2009)
11. Kranen, P., Kensche, D., Kim, S., Zimmermann, N., Müller, E., Quix, C., Li, X., Gries, T., Seidl, T., Jarke, M., Leonhardt, S.: Mobile mining and information management in healthnet scenarios. In: Proc. of the 9th IEEE MDM (2008)
12. Kranen, P., Seidl, T.: Harnessing the strengths of anytime algorithms for constant data streams. DMKD Journal, ECML PKDD Special Issue 2(19) (2009)
13. Lawder, J.: Calculation of mappings between one and n-dimensional values using the hilbert space-filling curves. Technical Report JL1/00 Birkbeck College, University of London (2000)
14. Leutenegger, S.T., Edgington, J.M., Lopez, M.A.: Str: A simple and efficient algorithm for r-tree packing. In: ICDE, pp. 497–506 (1997)
15. Seidl, T., Assent, I., Kranen, P., Krieger, R., Herrmann, J.: Indexing density models for incremental learning and anytime classification on data streams. In: EDBT/ICDT (2009)
16. Ueno, K., Xi, X., Keogh, E.J., Lee, D.-J.: Anytime classification using the nearest neighbor algorithm with applications to stream mining. In: ICDM (2006)
17. Vasconcelos, N., Lippman, A.: Learning mixture hierarchies. In: NIPS (1998)

Summarizing Multidimensional Data Streams: A Hierarchy-Graph-Based Approach

Yoann Pitarch, Anne Laurent, and Pascal Poncelet

LIRMM - Univ. Montpellier 2, CNRS
161 rue Ada, 34095 Montpellier, France
{pitarch,laurent,poncelet}@lirmm.fr

Abstract. When dealing with potentially infinite data streams, storing the whole data stream history is unfeasible and providing a high-quality summary is required. In this paper[1], we propose a summarization method for multidimensional data streams based on a graph structure and taking advantage of the data hierarchies. The summarization method considers the data distribution and thus overcomes a major drawback of the Tilted Time Window common framework. We adapt this structure for synthesizing frequent itemsets extracted on temporal windows. Thanks to our approach, as users do not analyze any more numerous extraction results, the result processing is improved.

1 Introduction

With the rapid development of information technology, many applications (web log analysis, medical equipment monitoring, etc.) have to deal with data streams. A data stream is defined as a potentially infinite sequence of precise and changing data arriving at an intensive rate. Due to the high-speed constraint, stream data can be read only once (one-pass constraint [1]) and storing the whole stream history is impossible. Nevertheless, decision makers need to analyze the data stream history, leading us to propose data stream summarization methods. As most of stream data are multidimensional and can be considered at multiple levels of precision (referred to as MD/MT data), providing an on-line multidimensional and multilevel analysis on such data streams would be interesting in order to make profit of the OLAP technology in static datawarehouses.

To the best of our knowledge, only two approaches profit from OLAP technologies for the MD/ML data stream summarization. In [2], the temporal dimension is compressed thanks to Tilted Time Windows [3] (TTW). The most recent history is registered at the finest granularity while the older history is registered at coarser granularity. The user habits are exploited to choose the materialized cuboids. In spite of an interesting architecture, the storage cost can be reduced. [4] overcame this drawback by introducing *precision functions* which define for each granularity level of every dimension the minimal interval of a TTW to avoid storing unqueried or computable data. Globally, the existing approaches focus

[1] Part of the MIDAS project funded by the French ANR (ANR-07-MDCO-008).

on which cuboids must be materialized but none of them reconsiders the use of TTWs. In spite of a good compression ratio, changing the time granularity at regular intervals can lead to an important loss of precision. Indeed, this mechanism does not take the data distribution into account. For instance, if an item rarely occurs in the stream, it could be useful to keep precise informations about its occurrences. However, with the TTW mechanism, this information would be lost after the first aggregation.

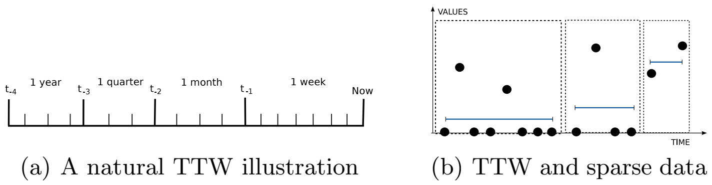
In this paper, we thus propose a graph-based framework for summarizing MD/ML data streams In this approach, if an item frequently appears in the stream, it is useless to conserve the precise history of its occurrences. Conversely, rare items are kept as conserving their precise occurrences could be useful for supporting decisions. Thanks to dynamic lists, aggregations are performed only if an item occurs in several close windows of the stream. On the contrary, non-close occurrences are kept during a significant period.

2 Problem Statement

MT/ML Data. Let $D = \{D_1, ..., D_M\}$ be a set of M dimensions. Every dimension D_i is defined over a (finite or not) set of values $Dom(D_i)$. Every dimension D_i can be considered at several levels of granularity, composing a hierarchy H_i where: max_i is the number of levels in H_i with $H_i^{max_i}$ the finest level and H_i^1 the coarsest. Note that for every dimension we consider a value *** which can be defined as *all the values*. We have $x \in Dom(D_i^j)$ if x is defined on the level H_i^j. For instance, a hierarchy of a geographic dimension D_{Geo} could be $H_{Geo} = \{H_1 = ALL, H_2 = Continent, H_3 = Country, H_4 = City\}$ and we have $France \in Dom(D_{Geo}^3)$. A (multidimensional) item t is then defined as $t = (d_1, ..., d_M)$ so that for every $i = 1...M$, $d_i \in Dom(D_i)$. t is said to be a *Lowest Level Item* (LLI) if $\forall d_i \in [1, M]$, $d_i \in Dom(D_i^{max_i})$. On the opposite, t is said to be a *Highest Level Item* (HLI).

Mining Multidimensional Items in Data Streams. A data stream $S = B_0, B_1, ..., B_n$ is an infinite sequence of batches (temporal windows), where every batch is associated with a timestamp t (denoted by B_t). A batch B_i is defined as a set of transactions appearing over the stream at the i^{th} time unit. In a MD data stream context, the support is defined as: $supp_{B_i}(X) = count(X)/|B_i|$ where $count(X)$ is the number of transactions of B_i in which X appears and $|B_i|$ the number of transactions in B_i.

Tilted Time Windows. In stream data analysis, users are usually interested in recent changes at a fine granularity, and in long term changes at coarse scale. *Tilted Time Window* [3] have thus been introduced, and the degree of coarseness depends on the application requirements and on how old the time point is (see Figure 1(a)). However, changing the time granularity at regular intervals can lead to an important loss of precision. For instance, Figure 1(b) shows an item i that occurs rarely: classical TTW would rapidly aggregate these occurrences.



(a) A natural TTW illustration (b) TTW and sparse data

Fig. 1. TTW: illustration and drawback

3 Raw Stream Data Summarization

The hierarchies associated to the dimensions compose the base-structure. These nodes are structural and do not store anything. As multidimensional items keep coming, nodes storing the summary at different levels of granularity are created, updated or deleted dynamically. To overcome the above-mentioned drawback of TTW, the history of each LLI is conserved in dynamic lists which store the precise occurrences. Thus, aggregations are not performed at regular time intervals but only when some conditions (e.g., temporal proximity between elements of the list) are validated. Higher granularity nodes store classical TTWs. Figure 2 displays a simple example. Due to the potentially infinite length of a stream,

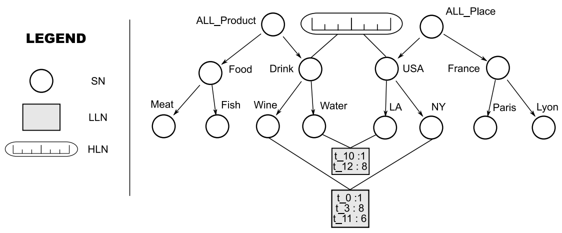


Fig. 2. Proposed structure. Item $(Water, LA)$ occurs once in the batch t_{10} and 8 times in batch t_{12}. More general items (e.g., $(Drink, USA)$) are kept in TTWs.

the accumulation of occurrences in dynamic lists is impossible. Mechanisms for aggregating or merging data are thus proposed:

1. If the same item appears in close temporal windows, they are merged and the result of this merging is propagated along the item generalizations.
2. A maximum size for each list is fixed. When a list reaches its maximal size, the oldest element is deleted.

3.1 Description of the Structure

Initially, the graph structure is composed by the dimension hierarchies. These nodes are called *structural nodes* SN. Histories of $LLIs$ are stored in nodes called the *Lowest Level Nodes* (LLN). Let $N_X = (X, Hist_X)$ be a LLN node so that X is a LLI and $Hist_X$ is a list containing pairs $< W : Count_W >$ where W is a time interval and $Count_W$ is the number of occurences of X_R in W.

If W represents more than one time unit, W_{beg} and W_{end} denote the bounds of the interval. Otherwise, the notation W is used. HLI are represented in our structure by *High Level Nodes* (HLN). Let $M_X = (X, T_X)$ be an HLN so that X is an HLI and T_X is a TTW storing the history of X.

3.2 Updating the Structure

A distance measure between LLI. Since stream data arrives at a very low level of granularity, the number of potential items can be huge. [2] proposes to tackle this problem by electing the lowest level of granularity which is interesting for the user (m-layer). Data are systematically aggregated to this level of granularity. Sometimes, users need to keep a track of precise data. In such a context, two items could be different but semantically close. For this purpose, we propose a hierarchy-based distance *dist*. Let $A = \{X_1^A, X_2^A, ..., X_N^A\}$ et $B = \{X_1^B, X_2^B, ..., X_N^B\}$ be two LLI. We define $dist(A, B)$ as:

$$dist(A, B) = 1 - \frac{\sum_{1 \leq j \leq N} \frac{1}{lv(NCA(x_1,x_2))^2}}{N}$$

where N is the number of dimensions, $lv(x)$ the level of granularity of x (with $lv(x) = 1$ if x is a sheet of the hierarchy) and $NCA(x_1, x_2)$ is the nearest common ancestor of x_1 and x_2. Two items A and B are semantically close if $dist(A, B) < distMax$ where $distMax$ is a user-defined threshold.

Example 1. *Let A=(Wine,Paris) and B=(Wine,Lyon) be two LLI. We have* $NCA(Wine, Wine) = IdProduct$ *(*$prox(Wine, WIne) = 1$*) and* $NCA(Paris, Lyon) = Country$ *(*$Prox(Paris, Lyon) = \frac{1}{2^2}$*). Thus* $dist(A, B) = 1 - \frac{1+0.25}{2}$.

When a node N_X (where N_X is an LLN) already exists, it must be updated if X reappears (in the batch t for instance). Indeed, the pair $< t, count_t >$ is to $Hist_X$. Due to the storage constraint, a merge mechanism is proposed.

The Merge Operation. If an item occurs in close time intervals, it is unecessary to keep all its occurrences. Merging these occurrences in a naive manner would perturb the propagation on the HLN. For instance, let us consider the LLN *(Wine,NY)* and the TTW displayed on Figure 2 where an agregation is performed every three time units. Considering the aggregated value of $[T_0; T_3]$, it cannot be inserted in the second window of the TTW because it overlaps the first two windows. Indeed, each value in the second window represents three time units (more generally, each value stored in a window k represents the aggregation of $W_1 \times ... \times W_{k-1}$ time units). So, a merging can be performed if and only if the impacted interval represents one temporal granularity of the TTW.

Merging pairs stored in a node N_X and propagating the aggregated values along the generalization of X is performed as follows. Firstly, the pair f arising from the merging is computed and the associated pairs are deleted from $Hist_X$. This process launches a propagation along the generalization of the concerned item. The nodes sharing the same generalization are sought. Every $Hist$ is scanned for locating entries to participate to the aggregation. A pair cannot participate to two different merge operations. So, every located pair is marked. Lastly, the aggregated value is inserted at the appropriate position in the TTWs corresponding to the generalizations of X.

Example 2. *Let us consider the example from Figure 2 and let us suppose that a merging has to be performed on the* (Water-LA) *node. The pairs* $< t_{10} : 1 >$ *and* $< t_{12} : 8 >$ *are aggregated. Then, nodes sharing the same generalization (i.e.,* (Drink-USA)*) are sought, retrieving the node* Wine-NY*. Its list contains* $< t_{11}, 5 >$*, which can participate to the aggregation.*

Limiting the Size of the Lists. The merge mechanism allows for compression of lists but is insufficient to guarantee that the structure fits in main memory. Additional methods must be proposed in order to avoid memory overflows. A merging is performed if the interval represents one temporal granularity of the TTW. But it is unrealistic to consider all the granularities. Let us suppose that the TTW displayed in Figure 1(a) is used. Considering the whole TTW for the merging mechanism implies that we can potentially wait for 1 year before any merging. Storing a so long history in each list is inconceivable. So, we introduce a user-defined numerical parameter, W_{MAX}, which means that the maximum size of the possibly merged interval is $W_1 \times ... \times W_{MAX} - 1$. Secondly, the merging mechanism is not sufficient to limit the number of elements stored in a list. In fact, it is not possible to determine the data distribution in a stream and, consequently, it is impossible to predict the number of merging operations. So, a user-defined numerical parameter, *MAX-SIZE*, is introduced.Since the *MAX-SIZE*th element of the list can be possibly merged in the future, we authorize *MAX-SIZE* $+(W_1 \times ... \times W_{MAX} - 1) - 1$ elements per list.

Example 3. *Let us consider the TTW from Figure 2, with* MAX-SIZE$= 3$ *and* $W_{MAX} = W_3$*. The maximum size of the list is then* $3 + (3 \times 3) - 1 = 11$*.*

General Update Algorithm. When updating an LLN, the size of *Hist* is evaluated and compared to *MAX-SIZE*. If the size is smaller than *MAX-SIZE*, we check in the list if a merge operation is possible. If necessary, a merge operation is performed. Otherwise, the pair is inserted at the end of the *Hist*. If the size of the *Hist* is greater than or equal to $MAX - SIZE$, we get the $MAX - SIZE^{th}$ element in *Hist* and we check if a merge operation is possible. Otherwise, we check if $t_c - t_m < (W_1 \times ... \times W_{MAX} - 1)$. This check allows us to verify if the $MAX - SIZE^{th}$ element could be merged in the future. Otherwise, the list is full and any element could be merged. The oldest pair is thus deleted.

4 Frequent Itemset Synthesis

Frequent itemsets extracted over temporal windows can be considered as an interresting data stream summarization technique. However, we discussed the difficulty for decision makers to analyze the numerous and independent set of results manually. In this section, the minor adjustments to perform in order to take into account such specific input are presented.

Storing frequent itemsets instead of items requires that dynamic lists (resp. TTWs) cannot be stored in LLN (resp. HLN). So, some definitions must be adapted. Indeed, LLN and HLN nodes are now considered as structural nodes and do not store any history. Likewise $LLIs$, the history of each $LLIS$ is stored in nodes named the *Lowest Level Itemset Nodes* ($LLISN$). Let $N_X = (X : Hist_X)$ be a $LLISN$ node so that X is a $LLIS$ and $Hist_X$ is a set of pairs $< W : Supp_W >$ where W is a time interval and $Supp_W$ is the support of X in W. If W represents more than one time unit, we note W_{beg} (resp. W_{end}) the beginning (resp. the end) of the interval. On the contrary, the notation W is used. $HLIS$ are stored in nodes called *Highest Level Itemset Nodes* ($HLISN$). Let $R = (X : T)$ be an HLR so that X is an HLI and T is a TTW.

Methods presented in Section 3 can easily transposed to the synthesis of frequent itemsets. Due to both the lack of space and the extreme proximity with the above-written algorithms, they are not given here.

5 Experiments

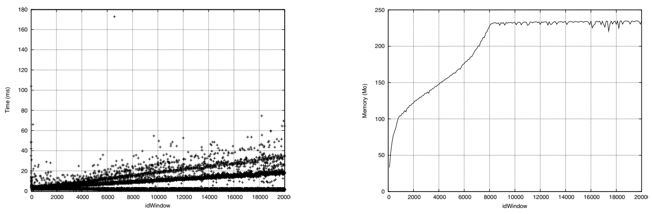
The feasibility of our approach is evaluated by considering the update time of the data structure and the main memory consumption. Experiments are conducted on a Intel(R) Xeon(R) CPU E5450@3.00GHz with 2GB of main memory, running Ubuntu 9.04. The methods are written in Java 1.6. We report and discuss here the most representative ones. Refer to our website[2] for complete results.

5.1 Synthetic Datasets

The data stream is simulated using a multidimensional random data generator (following a Random Uniform Distribution). D10L3C5W20T100SM10 stands for 10 dimensions, 3 granularity levels per dimension (except level *), node fan-out factor (cardinality) of 5 (i.e., 5 children per node), 20K temporal windows of 100 tuples each and proper-approach parameters of SIZE-MAX=10. Figure 3 presents a representative result obtained during the experimentations. On Figure 3(b), three distinct behaviors can be observed: quick increase of the memory consumption (no merging performed) then fair increase of the RAM consumption (occurrence of two concurrent phenomena: merging and filling up lists), and finally, stabilization of the memory usage because all the potential LN are created and insertions in lists are balanced by merging. Regarding the update time per window (Figure 3(a)), three distinct time scales can be noticed. The lowest one corresponds to a node creation or to a simple insertion in a list (performed almost instantaneously), the second one corresponds to merging and aggregation mechanisms (approximately 15ms) and the highest one is explainable by both insertion and merging (approximately 20ms).

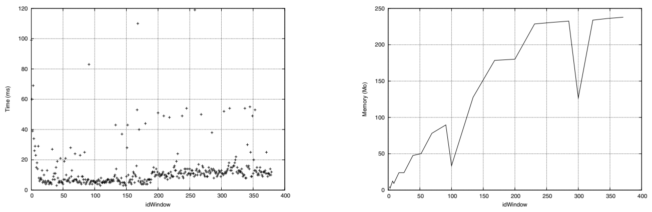
Due to the Random Uniform Distribution of data, paramaters which impact directly the number of potential items to store have a logical influence on both time and memory consumption performances. Indeed, the higher the number of

[2] `http://www.lirmm.fr/~pitarch/PAKDD10/experiments.html`



(a) Insertion time/window (b) RAM consumption

Fig. 3. Influence of the number of dimensions (L3C5W20T100SM10)



(a) Insertion time/window (b) Memory consumption

Fig. 4. Experiments conducted on frequent itemsets

items, the longer the time to perform a merge operation. Nevertheless, it can be noticed that performances become critical when the parameter values are extreme (e.g., when the depth of the hierarchies equals 7). In other experiments, results show the feasibility of the proposed method.

5.2 Real Dataset

We consider here industrial pumps transmitting physical informations (e.g., pressure, external temperature) over 10 dimensions. Hierarchies were arbitrarily built with the following characteristics. Every dimension has 3 levels of granularity and the average fan-out factor is 100. the dataset is dense. The input file is divided into windows containing 100 tuples.

Summarizing Items. We observe that memory is rapidly bounded as the dataset is very dense. The average insertion time is 50ms, and distinct time scales are observable. Moreover, the insertion time is relatively stable and this time is at worst 230ms.

Synthesizing Frequent Itemsets. Multidimensional itemsets and customer sequences were arbitrarily built. The average number of items per itemset is 25 and the average number of customers per client is 100. Then, a frequent itemset mining algorithm was applied[3] with a minSupp=10%. The average number of frequent itemsets per window is approximatively 100. Finally, we run our

[3] We use the implementation of FP-Growth provided by the Illimine project.

algorithm on those frequent itemsets. Figure 4(b) displays the results of memory consumption. The memory consumption stabilizes quickly (the frequent itemsets are almost the same on the whole data stream). The two off-peaks can be explained by the garbage collector. Regarding the insertion time, it can be noted that the simple insertions or list creations are a little slower than with items. This is explainable by the higher complexity of itemsets in comparison to items. Several merging and generalization mechanisms can also be observed.

6 Conclusion

In this paper, we tackle the problem of summarizing multidimensional and multilevel data stream thanks to a graph structure and provide efficient algorithm for updating this structure. Moreover, thanks to dynamic lists, we overcome the major drawback of the TTW: taking the data distribution into account. Finally, we show how frequent itemsets can be synthesized in order providing a comfortable solution for decision support. Our experiment study on both synthetic and real datasets shows that our summarization structure is efficient in both time and space, allowing us to consider numerous possible extensions.

References

1. Aggarwal, C.C.: Data Streams: Models and Algorithms. Advances in Database Systems (2007)
2. Han, J., Chen, Y., Dong, G., Pei, J., Wah, B.W., Wang, J., Cai, Y.D.: Stream cube: An architecture for multi-dimensional analysis of data streams. Distributed Parallel Databases 18(2) (2005)
3. Giannella, C., Han, J., Pei, J., Yan, X., Yu, P.: Mining frequent patterns in data streams at multiple time granularities (2002)
4. Pitarch, Y., Laurent, A., Plantevit, M., Poncelet, P.: Multidimensional Data Streams Summarization Using Extended Tilted-Time Windows. In: FINA (2009)
5. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proc. of the 20th Int. Conf. on Very Large Data Bases (VLDB'94), pp. 487–499 (1994)

Efficient Trade-Off between Speed Processing and Accuracy in Summarizing Data Streams

Nesrine Gabsi[1,2], Fabrice Clérot[1], and Georges Hébrail[2]

[1] France Télécom RD, 2, avenue P. Marzin 22307 Lannion, France
fabrice.clerot@orange-ftgroup.com
[2] Institut TELECOM, TELECOM ParisTech, 46 Rue Barrault 75013 Paris, France
{nesrine.gabsi,georges.hebrail}@enst.fr

Abstract. Data streams constitute the core of many traditional (e.g. financial) and emerging (e.g. environmental) applications. The sources of streams are ubiquitous in daily life (e.g. web clicks). One feature of these data is the high speed of their arrival. Thus, their processing entails a special constraint. Despite the exponential growth in the capacity of storage devices, it is very expensive - even impossible - to store a data stream in its entirety. Consequently, queries are evaluated only on the recent data of the stream, the old ones are expired. However, some applications need to query the whole data stream. Therefore, the inability to store a complete stream suggests the storage of a compact representation of its data, called summaries. These structures allow users to query the past without an explosion of the required storage space, to provide historical aggregated information, to perform data mining tasks or to detect anomalous behavior in computer systems. The side effect of using summaries is that queries over historical data may not return exact answers, but only approximate ones.

This paper introduces a new approach which is a trade-off between the accuracy of query results and the time consumed in building summaries.

1 Introduction

A data stream is an ordered, continuous sequence of timestamped data elements [8]. The information naturally occurs in the form of a sequence of data values; examples include sensor data, Internet traffic, financial tickers, transaction logs, etc. The potentially infinite nature of data streams and their high arrival rate imply an inability to store a data stream in its entirety and restrict queries and algorithms to process the data of a stream on the fly in a one pass fashion (i.e: without a prior data storage). In order to be processed in real time, queries must be specified before the beginning of streams. These queries run continuously over a period of time and incrementally return new results as new data arrive. The inability to store a complete stream suggests the use of approximate summary structures, referred to in the literature as synopses [9] or digests [3]. By definition, a summary is an incomplete representation of the historical data of a stream. Summarizing online data streams has been largely studied with several techniques like sketching, sampling, building histograms, and wavelets [2] [7] [11].

In this paper, we investigate using both sampling and clustering processes to make historical summaries on data streams. The sampling method is known to be fast and efficient for answering important classes of queries, but this technique does not give good results for historical queries [4]. The clustering process enables to maintain a good performance for queries applied on distant past. By taking advantages of the sampling and the clustering processes, we show in this paper that we can make a summary of the whole data stream. Through our performance measurements, we show that the data kept using our algorithm, allow us to have good results for queries that cover a very distant past.

In this paper, we are interested in querying the whole data stream specially, the historical part of the stream. We study the multi-dimensional and numerical data stream produced by a source 'F'. Each data stream can be viewed as a sequence of $< t_i, v_1, v_2, ..., v_n >$ where: $v_1, v_2, ...v_n$ represent the element values for each attribute and t_i is the time when the reading element was produced by the source 'F'. We assume that the elements come under increasing timestamps. The study of delay in the arrival of elements can be managed by our approach but it is not the purpose of this paper.

We are interested in aggregating the stream values within a time interval [6]. This means answering range queries. A range query is a pair Q $< Agg, [t_{start}, t_{end}] >$ where Agg represents an aggregate operation (sum, count, etc.) and $[t_{start}, t_{end}]$ are the bounds of the time period over which the aggregate is calculated. The aggregate operation is computed over the data stream's summary.

We present the Reservoir Hybrid (RH) approach in order to build historical summaries of data streams. It is a two stage summary: initially the summary is in the form of samples then, the elements of these samples evolve to make part of a cluster summary. This proposal offers a good compromise in terms of accuracy and run-time.

2 Related Work

In this section, we briefly describe the CluStream and StreamSamp approaches which are used to create the RH approach for summarizing data streams.

CluStream [1] is based on clustering of quantitative data but it can provide a structure particularly suited for the data stream summary. The objective is to build a summary as evolutionary micro-clusters including snapshots which are stored regularly. Aggarwal was inspired by the BIRCH algorithm [12] using the CFV structure (Cluster Feature Vector) to represent clusters and expands on this concept by adding temporal features. The CFV structure maintains statistical data that summarize all the elements of micro-clusters.

CluStream proceeds in two steps: the first one is the building of a data summary after the initialization of k micro-clusters (using the k-means algorithm). This step consists on the micro-clusters creation and maintenance. When a new stream element enters the Clustream process, its distance to the centroid of each micro-cluster is calculated and then assigned to the nearest micro-cluster. The CFV of this cluster is updated without storing the belonging of this element to the cluster. The second step is characterized by post-analysis which can be applied to the stored snapshots. At each clock time, the algorithm takes a snapshot and saves it according to a pyramidal time frame[1]. This

[1] Snapshots are stored at different levels of granularity. This structure favors recent time frames to older ones.

consists of storing on disk the CFV of all micro-clusters. For post-analysis purposes CFVs allow the subtraction of two snapshots. Approximation of results for statistical queries can be computed over the pre-specified time horizon.

CluStream keeps representative snapshots even for old stream elements. This allows the monitoring of the data stream over time. However, one weakness of the algorithm is that the process of distance calculations is expensive. A second limitation consists of the high number of parameters which depend on the nature of the stream and the arrival speed of elements.

StreamSamp [5] is based on sampling a data stream. It was developed to overcome the limitations of CluStream. It combines a memory-based reduction method which is random sampling and a time-based reduction system which is tilted windows [2]. The algorithm proceeds in two steps: the first one is the building of data summary by sampling and re-sampling stream elements. Upon arrival, the data are sampled in a purely random way with a fixed sampling rate, α, and placed in samples of size T. When T is reached StreamSamp, stores on the disk the sample elements and its starting and ending timestamps of constitution. The weight 1 is attributed to this sample. As it is impossible to keep all samples, when L samples of size T are filled, StreamSamp merges the two oldest samples following the tilted windows system. The weight of the resultant sample (created by random re-sampling) is multiplied by 2. It is a recursive operation. The second step allows the exploitation and analysis of the created summary. It consists of the exploration of the summary retained for a given period. The algorithm starts by constituting the final sample by concatenating elements having different weights to restore the flow of that period. Each element in this period is associated to its weight.

StreamSamp has the advantage of being fast on designing the data stream summary. However, its performance degrades over time because old elements increase in weight for a given sample size. Therefore, if a sample contains recent elements (much lower weight) and some old elements, the latter will increase the errors in the results of query answers.

3 Reservoir Hybrid Approach

The RH approach is presented as a trade-off between processing speed and accuracy in summarized data streams. The basic idea of this algorithm is that the elements from the data stream are first sent into StreamSamp to be processed. When the samples are no longer representative in terms of the two criteria detailed below, the sample elements are sent to CluStream. Depending on the chronological order, these elements are sent to the *reservoir* before sending to CluStream. Since it involves CluStream, only numerical data can be considered.

3.1 Representativeness of Samples

The representativeness of a sample is associated to two criteria: (i) the variance criterion and, (ii) the position criterion of the centroids. These criteria are based on inherent

[2] Summaries with a constant size are maintained covering time periods of varying sizes, shorter for the present and longer for the distant past.

features of the two processes: the random sampling for StreamSamp and the updating evolutionary micro-clusters for CluStream. In order to preserve a summary with good quality, we have to maintain a good quality for these two processes. The first criterion is checked for each attribute while the second one is checked considering all attributes together. The main idea is to transit from the StreamSamp process to CluStream based on a simultaneous check of these two criteria.

Variance Criterion. One of the steps in the StreamSamp process is random re-sampling. In this step, two samples E_1 and E_2, both of size N and covering respectively the time $[t_1 - t_2]$ and $[t_3 - t_4]$, are merged into a new sample E_3. The E_3 sample is created by randomly drawing N elements from E_1 and E_2 and covering the period $[t_1 - t_4]$. This step leads to a degradation of the summary. The variance criterion monitors random re-sampling and measures the 'quality' of the resulting sample.
Definition 1. A sample has a good quality if it can generate, from stored data, an approximate response to aggregate queries such as mean, variance, etc.

We aim to control the quality of the sample resulting from the merger of independent samples E_1 and E_2. We note that these samples have the same weight. The sample's quality is checked by controlling the accuracy of aggregate estimators. In this paper, we check the accuracy of the mean estimator noted $\overline{x}$. However, we could make this criterion more severe by controlling the quality of all used aggregates (mean, median, sum, etc). The goal of this criterion is to set a statistical bound on the mean estimator. Since a sample and random sampling are used, we know that with a confidence of 95%, we have the inequality:

$$\left|\overline{x} - \widehat{\overline{x}}(E_1 \cup E_2)\right| \le 1.96\sqrt{Var(\widehat{\overline{x}}(E_1 \cup E_2))}. \tag{1}$$

Were E_1 and E_2 are the two samples that have to be merged and $\widehat{\overline{x}}$ is the estimator of the mean.

The variance $Var(\widehat{\overline{x}}(E_1 \cup E_2))$ is estimated according to the following formula :

$$Var(\widehat{\overline{x}}(E_1 \cup E_2)) = (1 - \frac{2n}{N})(\frac{1}{2n})[\frac{1}{2n-1}\sum_{k \in E_1 \cup E_2}(x_k - \overline{x})^2]. \tag{2}$$

Where n is the sample size and N is the size of the involved population

To ensure that merger quality is satisfied, we define a threshold B (user defined parameter) that the error estimator must not exceed. The criterion is expressed using the inequality :

$$\frac{1.96\sqrt{Var(\widehat{\overline{x}}(E_1 \cup E_2))}}{\widehat{\overline{x}}(E_1 \cup E_2)} \le B \tag{3}$$

However, even if the criterion is met, we do not decide to merge, unless the second criterion, about the position of the centroids is checked.

Position Criterion of the Centroids. While the first criterion concerns the StreamSamp process, the position criterion of the centroids is related to the CluStream process. The random re-sampling process leads to a deterioration on the quality of the built summary.

This may cause a considerable change on the position of the centroids which will be calculated on the remaining samples. Consequently, the accuracy on the position of the centroids deteriorates. Therefore, like the first criterion, a minimum precision on the centroids position must be maintained by establishing a threshold over the distance between the centroids.

Unlike the classical approach of CluStream in which the algorithm processes the whole stream, in our CluStream version, the algorithm will only processes a sampled stream.

In order to maintain this accuracy, we check the centroid's precision at each re-sampling step. We calculate the distance between the centroid (G) (calculated from the samples to be merged (E_1 and E_2)) and the centroid ($\overline{G}$) (calculated from the estimated sample (E_3)). This distance must be below a threshold D. Otherwise, the required precision is no longer respected.

$$D = \epsilon \times \sum_{E_1 \cup E_2} (d^2(\overline{x}, x_i)) \tag{4}$$

Where ϵ is a user defined parameter fixed following the evolution of the centroids, and $\sum_{E_1 \cup E_2} (d^2(\overline{x}, x_i))$ is the intra-cluster inertia of the sample made up from ($E_1 \cup E_2$)

If one of these two criteria is no longer respected, the two corresponding samples will be handled by CluStream.

3.2 Insertion of the Samples in CluStream

The insertion of elements in CluStream depends on two parameters: (1) the weight of elements and, (2) the chronological insertion.

(1) To maintain the representativeness of the elements, the weight must be taken into consideration. Two strategies can be applied for the insertion of elements into CluStream: (i) each element i is inserted w_i times (w_i is the weight of the element i) or (ii) each element is multiplied by its weight and inserted once in the nearest micro-cluster. CluStream uses the Euclidean distance. It is not based on the correlation between attributes (i.e. Mahanalobis distance [10]). The two strategies offer the same results. Consequently, each element is inserted once because the first strategy needs $O(p * \tau * w_i)$ times to insert an element i into the nearest micro-cluster (with p the number of micro-clusters and τ the time needed to calculate the distance between an element and a micro-cluster).

(2) In CluStream, Aggarwal added a temporal extension to the classical form of CFV. In order to reflect the evolution of the stream data over a time period, it is necessary to insert elements in chronological order. Thus, during their move from StreamSamp to CluStream, data elements have to be processed according to the order of their respective timestamps: samples with higher weights must be moved away before the lower weighted samples.

Following the chronological order some samples have to be moved to CluStream only because they are older than another samples, although they still respect the merge criteria. The early transmission of samples from StreamSamp to CluStream leads to the following important drawbacks:

- Unnecessary waste of accuracy, especially for classification tasks.
- Unnecessary waste of time as the clustering process of CluStream is slower than sampling.
- StreamSamp empties quickly and once emptied we lose the performance of this algorithm (fast processing data stream, fast building summaries, good accuracy of the recent period, etc.).

To overcome these drawbacks, a buffer (referred to in the sequel as the *reservoir*) is introduced between StreamSamp and CluStream. As StreamSamp processes data faster than CluStream, the basic idea behind our proposal is to keep data in the StreamSamp process as long as possible, *i.e:* as long as the representativeness criteria are respected. The reservoir structure is filled with samples that: (i) do not satisfy the representativeness criteria and hence can not remain in the StreamSamp process; and (ii) can not be moved yet to the CluStream process because there are older samples which are still in the StreamSamp process. These samples are moved from the reservoir to CluStream when the storage space allocated is reached. The data transfer between StreamSamp, the reservoir and CluStream is based on two rules.

Rule for transmission to the reservoir. Let E_1, E_2 be the two oldest samples of a weight i in the StreamSamp process. Assume that L (the maximum number of samples of weight i) is reached and that E_1 and E_2 cannot be merged. In such case, we check an eventual merger between E_2 and E_3 (the third oldest sample of weight i). If this merger is possible, only E_1 is sent to the reservoir, and the samples E_2 and E_3 are merged. Otherwise, samples E_1 and E_2 are sent to the reservoir. As we cannot indefinitely send samples to the reservoir, we need to vacuum it dynamically by sending elements to CluStream.

Rule for transmission to the CluStream process. Once the reservoir is filled, the δ oldest samples are sent to the CluStream process (δ is a user defined parameter). These δ samples are jointly extracted from the StreamSamp process and from the reservoir. They are sent to CluStream in chronological order from the oldest to the newest. The storage space allocated for the reservoir is not predefined. Rather, as illustrated by the following formula, a global space is shared between the StreamSamp summary and the dynamic reservoir.

$$Size(Res) = Size(HSpace) - Size(SSamples) \tag{5}$$

Res: Reservoir, *HSpace*: total space allowed for the hybrid approach, *SSamples*: Samples in StreamSamp. Such a sharing mechanism allows a flexible management of the storage space. This is an important feature of our approach as the storage space required by StreamSamp from one hand and the reservoir from the other hand, highly data-dependent on the quality of the built samples. Indeed, if the merger's criteria are often satisfied, StreamSamp needs more space than the reservoir as few samples are sent to the reservoir. In the opposite case, the reservoir needs more storage space.

4 Empirical Results

We aim at assessing the performance of our algorithm and comparing it with CluStream and StreamSamp used on their own. To make the comparison fair, all algorithms use the same amount of memory to store their summaries. The algorithm parameters are presented in table 1. Real data sets KDD98[3] and CoverType[4] are used to evaluate the performance of the algorithms. In these evaluations, we are interested in the robustness and efficiency of algorithms for estimating queries which are evaluated over a time period that grows old over time. Thereby, we study the aging period [0-10000] at different timestamps ($t_{10000}, t_{20000}, etc.$). We repeat the StreamSamp and the RH approach 100 times because they include the sampling step. The result corresponds to the mean of these drawings.

Table 1. Parameters of Algorithms

StreamSamp	CluStream	RH approach
$\alpha = 1$	Nb of clusters = 50	$B = 0.25$
T = 200 elements by sample	Nb of snapshots by order = 32	$D = 5.10^{-4}$
L = 8 samples by order		$\delta = 2$

For reasons of limited space, we just present in this section the results for median as querying task and classification as data mining task. Furthermore, other kinds of analysis tasks have been applied (e.g. Mean, clustering) and present good results for the Reservoir Hybrid approach. Note that we also compared these algorithms with the classical Hybrid Approach (without using a reservoir), in order to study the impact of reservoir in the construction of the summary.

4.1 Median Evaluation

We study the performances of the different approaches on median estimation. The estimated error is calculated according to its ranking values:$error = \frac{|EstimatedRank - RealRank|}{WindowSize}$

The *Real Rank* is calculated over the original dataset (5000 in our case) while, the *Estimated Rank* is calculated over the resulted summary and the *Window Size* represents the number of elements studied for the median calculation(10000 in our case). The value of the estimated rank depends on the algorithm used to design the summary:

1. With StreamSamp, the estimated rank is easily calculated because the sampling process preserves the structure of elements. Firstly, all elements included on [0-10000] are extracted and sorted according to the attribute value. We choose the

[3] The dataset contains 95412 records and 481 attributes of information about people who have made charitable donations. After examining the stationarity of the stream, we use only 4 numerical attributes (from 54).

[4] The data contains 581012 elements and is defined by 54 variables of different types. Each element belongs to a class from 7 target classes. The goal is to predict the forest cover type from these variables.

element which divides the distribution into two equal parts. The estimated rank corresponds to the rank of this element in the original data set.

2. With the CluStream algorithm, stream elements are absorbed inside the micro-clusters. For that, on period [0-10000], we use the centroids of micro-clusters as values of elements and the weight corresponds to the number of elements in the micro-cluster. We extract the rank of the median value from the original data set. While the value may not be found (micro-cluster centroid), we search the rank of the nearest lowest value and the rank of the nearest highest value from the original data set. The estimated rank is the mean of these borders.
3. Using the RH approach or the classical Hybrid approach, it is possible to have the StreamSamp and the CluStream processes running in parallel. In this case, we extract from StreamSamp's summary and the reservoir, all elements included on [0-10000]. For CluStream, we search the closest snapshots kept between 0 and 10000 to extract the statistics. We merge the elements from StreamSamp's summary with the centroids of micro-clusters. Then, we calculate the estimated rank on this new set of data.

As shown in figure 1(a), the relative median error calculated on StreamSamp increases with the aging period [0-10000]. This error is calculated once on CluStream given that it keeps two snapshots. The RH approach adopts a similar behavior to StreamSamp for the recent periods. Then, when the quality of summary deteriorates it converges to an accuracy close to the CluStream behavior. This approach provides better performance than either summarizing approaches can provide separately and provides greater accuracy in estimating the median than the classical Hybrid approach.

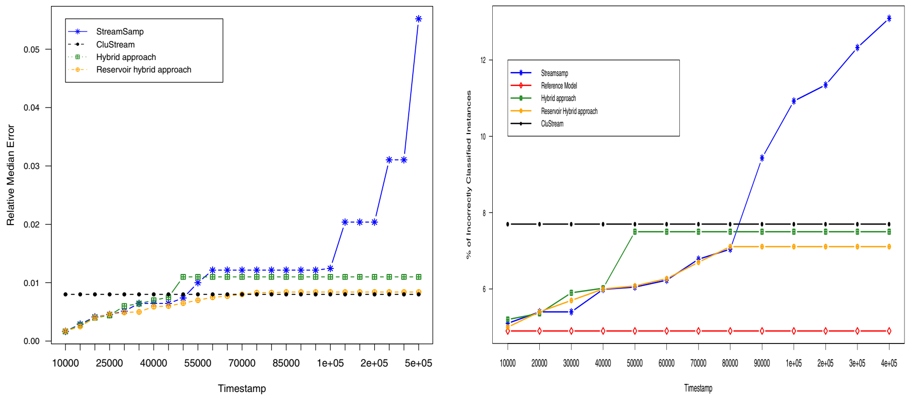


Fig. 1. a) Median Evaluation, b) Classification Evaluation

4.2 Classification Evaluation

We evaluate the performances of the generated models using the different summarizing algorithms. The models are constructed over the fixed period [0-10000] using the CoverType dataset. This dataset contains 7 labels, however, predicting 7 labels is difficult. To achieve this, we transform the data set to 2 labels: most frequent label (a majority), and all others. We compare the performance of the generated models with the reference which is constructed on the original dataset ([0-10000] in this task). The better algorithm had to develop a prediction model close to the reference. The classification evaluation is done on three steps:

1. Extraction of the summary on the period [0-10000]:
   - From StreamSamp built summary, we extract all the elements between timestamp 0 and 10000. The summary designed by StreamSamp has the advantage of retaining the same structure as the original data set.
   - The summary generated by CluStream only contains statistical data information, therefore, values and labels are absorbed in the micro-cluster. A pre-processing stage of data becomes necessary: (i) generating element values, (ii) adding label attribute.
     (i) For the first process, we use the information kept in micro-clusters to generate n_i elements (n_i is the number of elements in the micro-cluster i), following a Gaussian distribution. This operation is repeated 100 times because of randomly generation process. The result corresponds to the mean of these different drawings.
     (ii) For the second process, we have to associate each generated element to one label. To distinguish element belonging their labels, we use a binary coding in order to transform the 'n' labels in 'n' binary attributes to the dataset. Thereby, for each element, the value of the variable to predict is replaced by a binary value: '1' if value equals the variable and '0' otherwise. Due to this technique, we know for each micro-cluster the labels of the absorbed elements. The generated elements are associated according to these labels.
   - To extract the required part of summary built by the RH approach and the classical Hybrid approach, we pick up all samples from StreamSamp. Then, we concatenate this set of data by the elements generated from the CluStream micro-clusters as described above.
2. Construction of the model: The C4.5 algorithm is used on the extracted summaries in order to construct the models. However, other algorithms like CART or SVM can be applied.
3. Evaluation of the model: Models are evaluated using the training/test method.

As shown in figure 1(b), we compared the derived models constructed by algorithms to the reference model (without summarizing operations). StreamSamp built the closest model for recent periods because it used the real data unlike CluStream which uses data generated from micro-clusters. The classical Hybrid approach and the RH approach present better results in more recent periods since they used data from StreamSamp. For very distant past periods, the RH approach performances stabilize over time and presents the best results while, the StreamSamp performances continue to degrade.

4.3 Runtime Evaluation

In a data stream context, the runtime execution is a very important feature of processing stream data. We take account the global elapsed time for the data stream processing. We

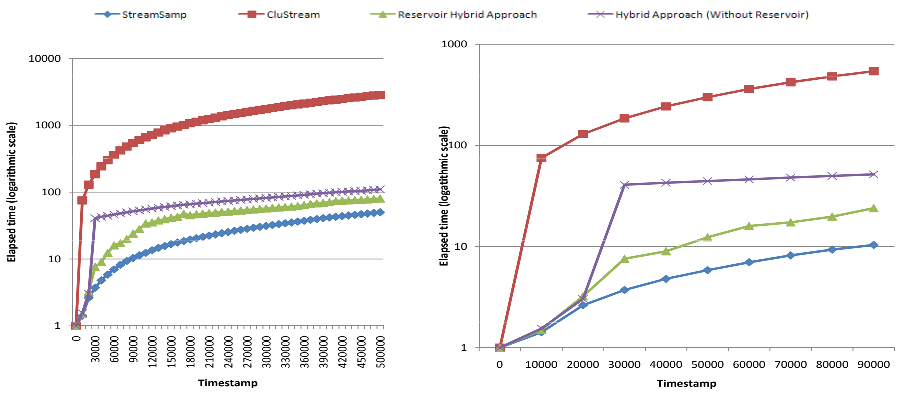


Fig. 2. a) Runtime Evaluation (logarithmic scale). b) Zoom on period [0-90000].

are not interested in the aging period [0-10000] but rather by the cumulative time for processing the whole data stream. As shown in figure 2(a) (logarithmic scale), StreamSamp provides the best performance and CluStream the worst one. The runtime performance of the RH approach remained between those of StreamSamp and CluStream. They are close to the StreamSamp's performance but still much faster than CluStream (more than 10 times faster than CluStream). StreamSamp algorithm uses only merging and sampling tasks. CluStream is the slowest because of updating operations of the CFV structures and the distance calculation between centroids.

The use of the reservoir provides a benefit in speed processing. The performance of the RH approach are better than the classical Hybrid approach and much better than CluStream. Furthermore, using the reservoir strategy, we avoid the heavy initialization step (Runtime evaluation Zoom in figure 2(b)).

4.4 Conclusion

Summarizing data streams is a difficult problem as we need to take into account two antagonistic problems: (i) the representativeness off kept data and hence, the accuracy of queries results; and (ii) the speed processing which is crucial in a data stream context. In this paper, we have developed an efficient method called *Reservoir Hybrid Approach* (RH approach) for summarizing data stream. We present the results for median and classification task, however, other kinds of queries (e.g. mean), and data mining tasks (e.g. clustering) was evaluated. All the evaluation results show that the RH approach solves the two antagonistic problems and provides best results. It provides a better speed-accuracy trade-off than existing approaches. The use of a reservoir makes summary building speed close to StreamSamp performances with an accuracy close to CluStream for distant past periods.

Future work includes the design of a query language allowing the exact querying of current data as well as the approximate querying of historical data.

References

1. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: VLDB, pp. 81–92 (2003)
2. Cormode, G., Garofalakis, M.N.: Sketching probabilistic data streams. In: SIGMOD Conference, pp. 281–292 (2007)
3. Cormode, G., Korn, F., Tirthapura, S.: Exponentially decayed aggregates on data streams. In: International Conference on Data Engineering, pp. 1379–1381 (2008)
4. Csernel, B.: Résumé généraliste de flux de données. PhD thesis, Ecole Nationale Supérieur des Télécommunications (February 2008)
5. Csernel, B., Clérot, F., Hébrail, G.: Streamsamp: Datastream clustering over tilted windows through sampling. In: ECML PKDD 2006 Workshop on Kbowledge Discovery from Data Streams (2006)
6. Cuzzocrea, A., Furfaro, F., Mazzeo, G.M., Sacca, D.: A grid framework for approximate aggregate query answering on summarized sensor network readings. In: Proc. of the 1st International Workshop on Grid Computing and its Application to Data Analysis (2004)
7. Gemulla, R., Lehner, W.: Sampling time-based sliding windows in bounded space. In: SIGMOD Conference, pp. 379–392 (2008)
8. Golab, L., Tamer Özsu, M.: Issues in data stream management. SIGMOD Record 32, 5–14 (2003)
9. Guha, S., Shim, K.: Offline and data stream algorithms for efficient computation of synopsis structures. In: VLDB '05, p. 1364. VLDB Endowment (2005)
10. Mahalanobis, P.C.: On the generalised distance in statistics. In: Proceedings National Institute of Science, India, April 1936, vol. 2, pp. 49–55 (1936)
11. Rueda, L.: An efficient algorithm for optimal multilevel thresholding of irregularly sampled histograms. In: SSPR/SPR, pp. 602–611 (2008)
12. Zhang, T., Ramakrishnan, R., Livny, M.: Birch: an efficient data clustering method for very large databases. SIGMOD Rec. 25(2), 103–114 (1996)

Subsequence Matching of Stream Synopses under the Time Warping Distance

Su-Chen Lin[1], Mi-Yen Yeh[2], and Ming-Syan Chen[1]

[1] Dept. of Electrical Engineering, National Taiwan University, Taipei, Taiwan
[2] Institute of Information Science, Academia Sinica, Taipei, Taiwan
sclin@arbor.ee.ntu.edu.tw, miyen@iis.sinica.edu.tw, mschen@cc.ee.ntu.edu.tw

Abstract. In this paper, we propose a method for online subsequence matching between histogram-based stream synopsis structures under the dynamic warping distance. Given a query synopsis pattern, the work continuously identifies all the matching subsequences for a stream as the histograms are generated. To effectively reduce the computation time, we design a Weighted Dynamic Time Warping (WDTW) algorithm which computes the warping distance directly between two histogram-based synopses. Our experiments on real datasets show that the proposed method significantly speeds up the pattern matching by sacrificing a little accuracy.

1 Introduction

Subsequence matching is a popular application in a data stream environment such as sensor network monitoring and financial data analysis. When given a query sequence, users would have great interests in continuously monitoring similar subsequences when a data stream keeps evolving. Therefore, a real-time and space-saving approach is required.

The similarity measurement is an important factor of the subsequence matching. Compared with the Euclidean distance, the dynamic time warping (DTW) [1] distance is more robust since it offers elastic scaling and shifting capabilities in time axis. To match two sequences of length M and N respectively, an intuitive solution is to compute the DTW distances of all possible matchings. However, the time complexity of this method is $O(MN^3)$ since it costs $O(MN)$ to obtain a DTW distance.

Various types of DTW algorithms on subsequence matching have been proposed [2,3]. However, it is impractical for stream applications to preprocess the whole data in advance as these methods did. Hence, Sakurai et al. [4] develop an online subsequence matching algorithm named SPRING, which is based on the DTW algorithm with relaxed boundary constraints. Given a query sequence of length M, SPRING spends $O(M)$ time to identify a matching subsequence at each data point of a stream. However, for a stream of length N, the total time cost of SPRING is $O(MN)$, which is heavy especially when N and M are large.

Searching for a better solution, we notice a growing interest in synopsis techniques [5] which meet the real-time requirement with a small accuracy loss in stream applications. Similarly, synopsis structures for speeding up the DTW algorithm are discussed. Keogh et al. proposed the PDTW algorithm which used a piecewise aggregate approximation (PAA) approach with equal-width histogram-based synopses to speeding up DTW [6]. In addition, synopsis structures with arbitrary-width histograms are designed for better approximation accuracy. Examples include the adaptive piecewise constraint approximation (APCA) [7] and the Haar wavelet reconstruction [8]. Chan et al. proposed a Haar wavelet-based approximation method under the time warping distance, but the method cannot deal with subsequences and has much overestimation [9].

For the above reasons, we present a new subsequence matching method under the dynamic time warping distance for data streams that are summarized with an arbitrary-width histogram-based synopsis structure. Each histogram contains a value and a timestamp indicating the end of this histogram as shown in Fig. 1(a). Given a query sequence Q with length M that summarized with m histograms, we want to continuously report subsequences of stream with distances to Q not greater than the threshold ϵ. We propose the Weighted Dynamic Time Warping (WDTW) algorithm that derives these matching subsequences. In order not to overestimate the warping distance, which could happen when one histogram of a stream synopsis matches multiple histograms of the other one, we have designed a method to lower the overestimated distance. The WDTW algorithm is shown to have $O(m)$ complexity in both time and space at the coming of each histogram of the stream. After processing n histograms, the total time complexity of our method is $O(mn)$, where $m \ll M$ and $n \ll N$ for synopsis streams.

To evaluate the WDTW algorithm, we conduct two experiments using a real dataset of time series. For comparisons, we implemented two other subsequence matching methods, referred to as Synopsis-DTW and MicroCell-DTW. The experimental results show that, when compared with MicroCell-DTW, our method and Synopsis-DTW have far low computational time costs. However, our method has only a little trade-off in accuracy, which is not true for Synopsis-DTW.

2 Preliminaries

Dynamic Time Warping (DTW) is a widely used distance measurement in time series applications. It can compute the distance between two series of different lengths since it solves the problem of shifting and scaling in the time axis. In essence, the DTW distance between two series $X = \{x_i | 1 \leq i \leq N\}$ and $Y = \{y_j | 1 \leq j \leq M\}$ is computed as follows. First, the distance between two data points is defined as $d(i,j) = \|x_i - y_j\|$. If x_i matches y_j, we define it as a *matching pair* (i,j). A sequence of all matching pairs from (1,1) to (N,M) between series X and Y is called a warping path W. Then, a warping distance for the path W is $Distance(W) = \sum_{(i,j) \in W} \|x_i - y_j\|$.

Obviously, there exists multiple warping paths between X and Y. The DTW distance between X and Y is defined as the warping path with the smallest

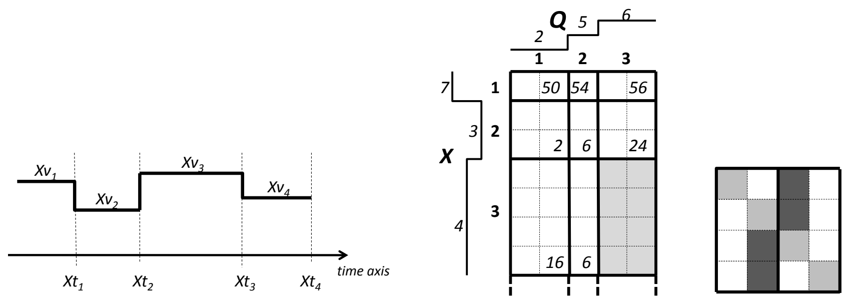


Fig. 1. (a) Histogram-based synopsis stream, (b) Accumulated distance matrix, (c) The dark shaded micro cells are overestimated distance area in directly accumulating distance

$Distance(W)$, which we defined as the optimal warping path. The dynamic programming technique can solve the optimal warping path problem in $O(MN)$ time complexity. Please refer to [1] for more details.

3 Subsequence Matching over Stream Synopses

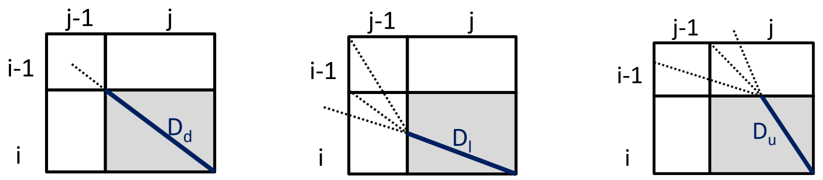
3.1 Problem Definition

Given a data stream of length N, an online stream is summarized as a sequence of histograms $X = \{x_1, x_2, .., x_n\}$, each of which has a height xv_i and an endpoint xt_i as shown in Fig. 1(a). We can denote a stream synopsis as $X = \{\langle xv_1, xt_1\rangle, ..., \langle xv_n, xt_n\rangle\}$. $X[t_s : t_e]$ is defined as a synopsis subsequence which starts from time t_s and finishes at t_e, where both t_s and t_e have to be endpoints of histograms of X. For ease of exposition, we denote a synopsis subsequence as $X[[i] : [j]]$ to mean that it starts from the i^{th} to the j^{th} histograms of X. Given the notations, we now define the subsequence matching problem.

Definition 1 (Synopsis Subsequence Matching). *Given an online running stream synopsis X, a query synopsis subsequence Q, and a threshold ϵ, the goal of synopsis subsequence matching is to locate all the subsequences $X[t_s : t_e]$ that satisfy $Dtw(X[t_s : t_e], Q) \leq \epsilon$*

3.2 WDTW: A Weighted Algorithm for Dynamic Time Warping

To solve the above issue, we propose *Weighted Dynamic Time Warping* (WDTW) method. When the distance between the stream synopsis $X = \{x_1, x_2, ..., x_n\}$ and the query sequence $Q = \{q_1, q_2, ..., q_m\}$ is derived, an accumulated distance matrix of $n \times m$ *cells* will be created to keep the histogram mapping information sequentially. As Fig. 1(b) shows, each cell is divided by the bold-solid lines according to width of each histogram. The cell (i, j) is constructed by the i^{th}



(a) Case1: diagonal cell (b) Case2: left cell (c) Case3: up cell

Fig. 2. Three alternatives for each matching procedure

histogram of X and j^{th} histogram of Q. The size of each cell is different due to the different width of each histogram. Therefore, each cell can be further divided into multiple square *micro cells* with side lengths equal to one time unit. For example, in Fig. 1(b), the shaded cell $(3, 3)$ contains 4×2 micro cells. Later we will show that in fact only the latest two rows of these cells need to be kept.

Similar to DTW, WDTW also works in a dynamic programming way. Intuitively, the warping distance at each cell is directly derived from three neighbor cells of various sizes in the matching procedure. However, this will result in overestimated distance. Using Fig. 1(c) as an example, if we directly compute the accumulating warping distance of the cell (3,3) from the cell (3,2), all the light and dark shaded micro cells will be counted in. In fact, the distance contributed by the dark shaded micro cells are redundant and should be eliminated.

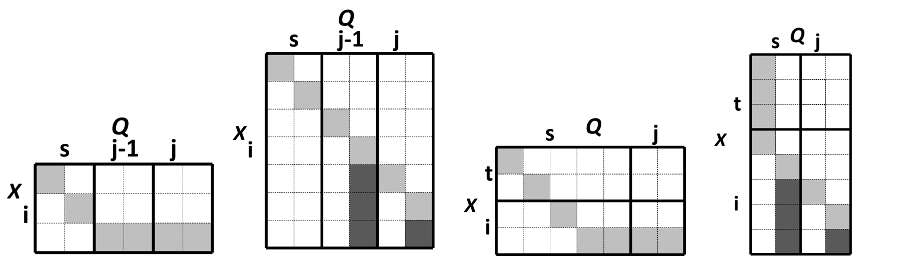
To lower the overestimated distance, our WDTW works as follows. First, we define $D(i, j)$ as the minimum accumulated distance between $X[[s] : [i]]$ and $Q[[1] : [j]]$, $s = 1, 2, ..., i$,

$$D(i,j) = \begin{cases} 0, & \text{if } j = 0, \\ \infty, & \text{if } i = 0, j \neq 0, \\ \min\{D_d(i,j), D_l(i,j), D_u(i,j)\}, & \text{otherwise.} \end{cases} \quad (1)$$

where $D_d(i, j)$, $D_l(i, j)$, and $D_u(i, j)$ denote the minimum distances for cell (i, j) with warping paths through the cell $(i-1, j-1)$, $(i, j-1)$, and $(i-1, j)$ respectively. We now discuss how to compute these distances case by case.

Case 1: Minimum Distance Path from the Diagonal Cell
In this case, the warping path comes from the cell $(i-1, j-1)$ to the current cell (i, j) as Fig. 2(a) shows. The path implies that the histogram x_{i-1} matches the histogram q_{j-1} and x_i matches q_j. The distance $D_d(i, j)$ can de obtained from sum of $D(i-1, j-1)$ and the distance of the cell (i, j).

The sub-optimal path of the current cell (i, j) passes $\max\{lx_i, lq_j\}$ micro cells, where lx_i and lq_j are the length of histogram x_i and q_j respectively. The micro cells passed by the sub-optimal path are called *steps* in the rest of this paper. Consequently, we can obtain the accumulated distance $D_d(i, j) = D(i-1, j-1) + d_{i,j} \times \max\{lx_i, lq_j\}$, where $d_{i,j} = \|xv_i - qv_j\|$. For example in Fig.1(b), the $D(3, 3)$ in the shaded area can be computed as: $D_d(3, 3) = D(2, 2) + d_{3,3} \times \max\{lx_3, lq_3\} = 6 + (4-6)^2 \times 4 = 22$. ■



(a) $lx_i \leq \sum_{k=s}^{j} lq_k$ (b) $lx_i > \sum_{k=s}^{j} lq_k$ (c) $E_v(i, j-1) = 0$ (d) $E_v(i, j-1) \neq 0$

Fig. 3. (a)(b) are two kinds of cell combinations, and (c)(d) are examples used in case 2. The shaded areas are passed by warping paths and the dark shaded areas stand for E_v for each cell.

Case 2: Minimum Distance Path from the Left Cell
The warping path from the cell $(i, j-1)$ to cell (i, j), as shown in Fig. 2(b), denotes that x_i matches the consecutive histograms q_{j-1} and q_j. We use a combined calculation of the consecutive histograms in a row to lower the overestimated distance of $D_l(i, j)$. Without loss of generality, we assume that the start histogram of these consecutive ones is q_s. Our major insight is that the consecutive histograms that match the same histogram x_i can be combined. In other words, the consecutive cells (i, s) to (i, j), where $s < j$, will be considered as a cell combination, where the sub-optimal warping path Ω goes from the upmost-leftmost micro cell of (i, s) to the bottommost-rightmost micro cell of (i, j).

Definition 2 (The Adjustable Distance in the Vertical Direction). *For the warping path passes from the cell (i, s) to the cell (i, j), where s is the smallest index of these consecutive ones as a combination, the value $E_v(i, j)$ is defined to represent the adjustable distance of cell (i, j) in the vertical direction as follows.*

$$E_v(i,j) = \begin{cases} 0 & ,if\ lx_i \leq \sum_{k=s}^{j} lq_k \\ \min\{d_{i,s}, ..., d_{i,j}\} \times (lx_i - \sum_{k=s}^{j} lq_k) & ,otherwise \end{cases} \tag{2}$$

Lemma 1. *The total distance of the sub-path, in a cell combination from (i, s) to (i, j), of any warping path that contains it is: $\sum_{k=s}^{j} (d_{i,k} \times lq_k) + E(i, j)$.*

Proof. Based on case 1, the sub-optimal path of the cell combination passes $\max\{lx_i, \sum_{k=s}^{j} lq_k\}$ steps (micro cells). If $\sum_{k=s}^{j} lq_k \geq lx_i$ as Fig. 3(a) shows, the distance of the sub-path would be $\sum_{k=s}^{j} (d_{i,k} \times lq_k)$ and $E_v(i, j)$ would be 0. If $lx_i > \sum_{k=s}^{j} lq_k$ as Fig. 3(b) shows, in addition to the diagonal steps, the path has to pass further $lx_i - \sum_{k=s}^{j} ly_k$ vertical steps. In order to obtain the optimal path, the vertical steps must be in the cell which has minimum $d_{i,k}$, where k is from s to j. In this condition, the adjustable distance in vertical direction is $d_{min} \times lx_i - \sum_{k=s}^{j} ly_k = E_v(i, j)$, which is the dark shaded area in Fig. 3(b) for example. Therefore, the total distance of the sub-path is $\sum_{k=s}^{j} (d_{i,k} \times lq_k) + E_v(i, j)$. □

In sum, Lemma 2 shows how to calculate the value of $D_l(i,j)$.

Lemma 2. *The accumulated distance $D_l(i,j)$, where the path comes from the cell $(i,j-1)$, can be obtained with following equation in O(1) time.*

$$D_l(i,j) = D(i,j-1) - E_v(i,j-1) + d_{i,j} \times lq_j + E_v(i,j) \tag{3}$$

Proof. Without loss of generality, for the sub-path from the cell (t,s) to the cell (i,j), where $t \leq i$ and $s < j$, there are two cases of $E_v(i,j-1)$. When $E_v(i,j-1) = 0$ as Fig. 3(a) and 3(c) show, no adjustable distance in the vertical direction should be distributed to the next cell. Hence, $E_v(i,j)$ is also equal to 0, and Eq. (3) is obtained in this case.

When $E_v(i,j-1) \neq 0$, as the dark shaded area shown in Fig. 3(b) and 3(d), the vertical steps in the previous cells can be distributed to the cell (i,j). With Lemma 1 and Definition 2, $D_l(i,j)$ can be obtained as follows.

$$\begin{aligned} D_l(i,j) &= D(t-1,s-1) + \sum_{k=t}^{i-1} (d_{k,s} \times lx_k) + \sum_{k=s}^{j} (d_{i,k} \times lq_k) + E_v(i,j) \\ &= D(i,j-1) - E_v(i,j-1) + d_{i,j} \times lq_j + E_v(i,j) \end{aligned} \quad \square$$

Based on Lemma 2, the information required to calculate $D_l(i,j)$ is only stored in cell (i,j) and $(i,j-1)$. Since the computation time of $E_l(i,j)$ and $E_l(i,j-1)$ is constant, $D_l(i,j)$ is obtained in constant time. We give an example of this case. In Fig. 1(b), $E_v(3,3) = \min\{d_{3,2}, d_{3,3}\} \times (lx_3 - lq_2 - lq_3) = 1 \times (3-2) = 1, D_l(3,3) = D(3,2) - E_v(3,2) + d_{3,3} \times lq_3 + E_v(3,3) = 6 - 3 + 4 \times 2 + 1 = 12.$ ■

Case 3: Minimum Distance Path from the Up Cell
In this case, the warping path passes through cell $(i-1,j)$ as Fig.2(c) shows. Since this case is similar to case 2 despite the consecutive cells are in a column, $D_u(i,j)$ can also be derived in constant time. In Fig. 1(b), $E_h(3,3) = 0$, $D_u(3,3) = D(2,3) - E_h(2,3) + d_{3,3} \times lx_3 + E_h(3,3) = 24 - 0 + 1 \times 4 + 0 = 28.$ ■

After deriving the distance values of the three cases, the minimum accumulated distance can be obtained. Continuing the previous example, $D(3,3) = \min\{D_d(3,3), D_l(3,3), D_u(3,3)\} = \min\{22, 12, 28\} = 12$.

We now describe how WDTW identifies matching subsequences. For each synopsis histogram x_i arriving at time t_i, WDTW computes the i^{th} row of the accumulated distance matrix $D(i,j)$ where $j = 1, 2, ..., m$. Then, the warping distance of the most similar subsequence, $X[t_s : t_i]$, to Q is $Dtw(X[t_s : t_i], Q) = D(i,m)$. To get the start time t_s, each cell (i,j) keeps the start time index where the warping distance comes from in the matrix $S(i,j)$. Therefore, when constructing the warping path, the start and end time of the most similar subsequence are kept in the last cell (i,m) of this path. For example, in Fig. 1(b), $t_s = S(3,2) = S(2,1) = t_2$. Hence, $X[t_2 : t_3]$ is reported at t_3 if $Dtw(X[t_2 : t_3], Q) \leq \epsilon$.

Notice that as we compute $D(i,j)$, $S(i,j)$ is also obtained. Since the computation cost of each $D(i,j)$ and $S(i,j)$ is $O(1)$, the computation time of m histograms is $O(m)$. Also, only the information of cell$(i-1,j-1)$,$(i,j-1)$, and $(i-1,j)$ is used. Therefore, WDTW only needs to keep the latest two rows of cells, i.e., $O(2 \times m) = O(m)$ in space.

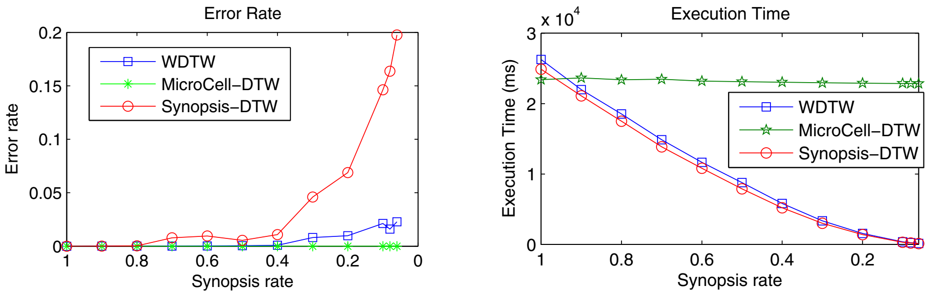


Fig. 4. Error rate and execution time at different synopsis rates

4 Performance Evaluation

4.1 Experiment Setup and Performance Metrics

The real datasets are downloaded from the UCR Time Series Classification / Clustering Archive [10]. We choose time series in the *posture* dataset. The first pattern in each series was used as the query patterns with length $M = 1024$, while the rest part of the series as data streams with length $N = 32768$. The synopsis histograms in the following experiments were built by Haar wavelet decomposition with varied synopsis rate [8], which is defined as a ratio of the number of histograms to the number of points in the original stream.

We compared our algorithm with two methods: MicroCell-DTW and Synopsis-DTW. MicroCell-DTW divided each histogram into multiple one-time-unit histograms. It computed the warping distance based on micro cells, not the cells, and the matching problem was solved using the SPRING method [4]. On the other hand, Synopsis-DTW computed warping distance directly on the cells as WDTW did, but was regardless of handling the overestimated distance.

4.2 Experimental Results

The first experiment examined the accuracy of WDTW and Synopsis-DTW. The subsequences produced by MicroCell-DTW were regarded as the benchmark since it produced the correct warping distance based on the synopsis histograms. The error rate is defined as the ratio of the sum of false alarms and misses to the number of correct subsequences and the reported subsequences by either WDTW or Synopsis-DTW. For each missed or false alarmed subsequence, the penalty is the ratio of the time interval of the false alarmed/missed part to the whole length of itself.

The results were shown in Fig. 4(a). When the synopsis rate decreased, the error rate of Synopsis-DTW increased significantly. In contrast, the error rate of WDTW increased much slightly. For example, the error rate of WDTW is 2.2% while that of Synopsis-DTW is up to 19.7% when the synopsis rate is 0.06. This shows the importance of dealing with the overestimated distance.

The second experiment examined the speed of the three algorithms. The results are in Fig. 4(b). MicroCell-DTW was independent of the synopsis rate since it examined each micro cells of a stream, where the number is equal to the original stream length. Under the synopsis rate of 0.06, MicroCell-DTW can only process 0.85 histograms each millisecond. In contrast, WDTW can process 141 histograms under the same synopsis rate, which is 165 times faster than MicroCell-DTW did. In other words, WDTW processed the online subsequence matching 165 times faster than MicroCell-DTW did. On the other hand, WDTW and Synipsis-DTW had almost the same computation cost, which decreased along with the synopsis rates. This shows that the processing time of the overestimated distance is very small. Concluded from Fig. 4(a) and 4(b), WDTW processes the online subsequence matching efficiently while sacrificing only a very little accuracy.

5 Conclusion

We presented WDTW, an efficient online subsequence matching algorithm under dynamic time warping in a streaming environment. Once a various-width synopsis histogram of a stream is generated, according to the query sequence, WDTW reports all the matching subsequences. The experimental results show that when compared with MicroCell-DTW and Synopsis-DTW, WDTW has a low computation cost to meet the time and space constraints of streams, with a little trade-off in accuracy.

References

1. Sakoe, H.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing (1978)
2. Chen, Y., Nascimento, M.A., Chin, B., Anthony, O., Tung, K.H.: Spade: On shape-based pattern detection in streaming time series. In: Proc. of ICDE (2007)
3. Han, W.S., Lee, J., Moon, Y.S., Jiang, H.: Ranked subsequence matching in time-series databases. In: Proc. of VLDB (2007)
4. Sakurai, Y., Faloutsos, C., Yamamuro, M.: Stream monitoring under the time warping distance. In: Proc. of ICDE (2007)
5. Aggarwal, C.C., Yu, P.S.: A survey of synopsis construction in data streams. Advances in Database Systems (2009)
6. Keogh, E., Pazzani, M.: Scaling up dynamic time warping for datamining applications. In: Proc. of the SIGKDD (2000)
7. Keogh, E., Chakrabarti, K., Mehrotra, S., Pazzani, M.: Locally adaptive dimensionality reduction for indexing large time series databases. In: Proc. of SIGMOD (2001)
8. Burrus, C.S., Gopinath, R.A., Guo, H.: Introduction to wavelets and wavelet transforms: A primer. Prentice Hall, Englewood Cliffs (1997)
9. Chan, F.K.P., chee Fu, A.W., Yu, C.: Haar wavelets for efficient similarity search of time-series: with and without time warping. In: IEEE TKDE (2003)
10. Keogh, E., Xi, X., Wei, L., Ratanamahatana, C.A.: Ucr time series classification/clustering archive, http://www.cs.ucr.edu/~eamonn/time_series_data/

Normalized Kernels as Similarity Indices

Julien Ah-Pine

Xerox Research Centre Europe
6 chemin de Maupertuis
38240 Meylan, France
julien.ah-pine@xrce.xerox.com

Abstract. Measuring similarity between objects is a fundamental issue for numerous applications in data-mining and machine learning domains. In this paper, we are interested in kernels. We particularly focus on kernel normalization methods that aim at designing proximity measures that better fit the definition and the intuition of a similarity index. To this end, we introduce a new family of normalization techniques which extends the cosine normalization. Our approach aims at refining the cosine measure between vectors in the feature space by considering another geometrical based score which is the mapped vectors' norm ratio. We show that the designed normalized kernels satisfy the basic axioms of a similarity index unlike most unnormalized kernels. Furthermore, we prove that the proposed normalized kernels are also kernels. Finally, we assess these different similarity measures in the context of clustering tasks by using a kernel PCA based clustering approach. Our experiments employing several real-world datasets show the potential benefits of normalized kernels over the cosine normalization and the Gaussian RBF kernel.

Keywords: Kernels normalization, similarity indices, kernel PCA based clustering.

1 Introduction

Measuring similarity between objects is a fundamental issue for numerous applications in data-mining and machine learning domains such as in clustering or in classification tasks. In that context, numerous recent approaches that tackle the latter tasks are based on kernels (see for example [1]). Kernels are special dot products considered as similarity measures. They are popular because they implicitly map objects initially represented in an input space, to a higher dimensional space, called the feature space. The so-called kernel trick relies on the fact that they represent dot products of mapped vectors without having to explicitly represent the latter in the feature space. From a practical standpoint, kernel methods allow one to deal with data that are not easy to linearly separate in the input space. In such cases, any clustering or classification method that makes use of dot products in the input space is limited. By mapping the data to a higher dimensional space, those methods can thus perform much better. Consequently, many other kinds of complex objects can be efficiently treated by using kernel methods. We have mentioned previously that kernels are generally introduced as similarity measures but as underlined in [2], dot products in general do not necessarily fit one's intuition of a similarity index. Indeed, one could find in the literature several axioms that clarify the definition of a similarity index and in that context, any kernel does not necessarily satisfy all of them. As an example, one of these conditions that a dot product, and thus a kernel, does not always respect, is the maximal self-similarity axiom which states that the object to which any object should be the most similar, is itself.

In this paper, we are interested in designing kernels which respect the basic axioms of a geometrical based similarity index. In that context, kernels normalization methods are useful. Basically, the most common way to normalize a kernel so as to have a similarity index, is to apply the cosine normalization. In that manner, maximal self-similarity for instance, is respected unlike for unnormalized kernels. In this work we propose a new family of kernel normalization methods that generalizes the cosine normalization. Typically, the cosine normalization leads to similarity measures between vectors that are based upon their angular measure. Our proposal goes beyond the cosine measure by refining the latter score by using another geometrical based measure which relies on the vectors' norm ratio in the feature space. We give the following example in order to motivate such normalized kernels. Let us take two vectors which are positively colinear in the feature space. In that case, their cosine measure is 1. However, if their norms are not the same ones therefore, we cannot conclude that these two mapped vectors are identical. Accordingly, their similarity measure should be lower than 1. Unlike the cosine normalization, the normalization approaches that we introduce in this paper aim at taking into consideration this point.

The rest of this paper is organized as follows. In section 2, we formally introduce new normalization methods for kernels. Then, in section 3, we give several properties of the resulting normalized kernels in the context of similarity indices. We show that using normalization methods allows one to make any kernel satisfy the basic axioms of a similarity index. Particularly, we prove that these kernel normalizations define metrics. In other words, we show that normalized kernels are kernels. In section 4, we illustrate the benefits of our proposal in the context of clustering tasks. The method we use in that regard, relies on kernel PCA based k-means clustering which can be understood as a combination between kernel PCA [3] and k-means via PCA [4]. This two step approach is a spectral clustering like algorithm. Using several datasets from the UCI ML repository [5], we show that different normalizations can better capture the proximity relationships of objects and improve the clustering results of the cosine measure and of another widely used normalized kernel, the Gaussian Radial Basis Function (RBF) kernel. We finally conclude and sketch some future works in section 5.

2 Kernels and Normalization Methods

We first recall some basic definitions about kernel functions and their cosine normalization. We then introduce our new kernel normalization methods.

2.1 Kernel Definition and the Cosine Normalization

Let denote $\mathcal{X}$ the set of objects, represented in an input space, that we want to analyze.

Definition 1 ((Positive semi-definite) Kernel). *A function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a positive semi-definite kernel if it is symmetric, that is, $K(\mathbf{x}, \mathbf{y}) = K(\mathbf{y}, \mathbf{x})$ for any two objects $\mathbf{x}, \mathbf{y}$ in $\mathcal{X}$ and positive semi-definite, that is:*

$$\sum_{i=1}^{n} \sum_{i'=1}^{n} c_i c_{i'} K(\mathbf{x}_i, \mathbf{x}_{i'}) \geq 0 \tag{1}$$

for any $n > 0$, any choice of n objects $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathcal{X}$ and any choice of any numbers $c_1, \ldots, c_n$ in $\mathbb{R}$.

In the sequel, we will simply use the term kernel instead of positive semi-definite kernel. We have the following well-known property.

Theorem 1. *For any kernel K on an input space $\mathcal{X}$, there exists a Hilbert space $\mathcal{F}$, called the feature space, and a mapping $\phi : \mathcal{X} \rightarrow \mathcal{F}$ such that for any two objects $\mathbf{x}, \mathbf{y}$ in $\mathcal{X}$:*

$$K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle \tag{2}$$

where $\langle ., . \rangle$ is the Euclidean dot product.

When the objects of $\mathcal{X}$ are vectors represented in an input space which is an Euclidean space then, we can mention the following two well-known types of kernel functions:

– Polynomial kernels: $K_p(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + c)^d$, with $d \in \mathbb{N}$ and $c \geq 0$.
– Gaussian RBF kernels: $K_g(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right)$, where $\|.\|$ is the Euclidean norm and $\sigma > 0$.

After having recalled basics about kernels, we recall the definition of the cosine normalization of a kernel K, denoted K^0.

$$K^0(\mathbf{x}, \mathbf{y}) = \frac{K(\mathbf{x}, \mathbf{y})}{\sqrt{(}K(\mathbf{x}, \mathbf{x})K(\mathbf{x}, \mathbf{y}))} \tag{3}$$

Since we have $K(\mathbf{x}, \mathbf{x}) = \|\phi(\mathbf{x})\|^2$, it is easy to see that:

$$K^0(\mathbf{x}, \mathbf{y}) = \langle \frac{\phi(\mathbf{x})}{\|\phi(\mathbf{x})\|}, \frac{\phi(\mathbf{y})}{\|\phi(\mathbf{y})\|} \rangle$$

Moreover, we have $K^0(\mathbf{x}, \mathbf{x}) = 1$ for all objects in $\mathcal{X}$. This means that the objects in the feature space are projected on an unit hypersphere. In addition, we have the following geometrical interpretation from the feature space representation viewpoint:

$$K^0(\mathbf{x}, \mathbf{y}) = \cos(\theta(\phi(\mathbf{x}), \phi(\mathbf{y}))) \tag{4}$$

with $\theta(\phi(\mathbf{x}), \phi(\mathbf{y}))$ being the angular measure between the vectors in the feature space. In order to simplify the notations, we will denote θ for $\theta(\mathbf{x}, \mathbf{y})$ and $\cos\theta$ for $\cos(\theta(\phi(\mathbf{x}), \phi(\mathbf{y})))$, thereafter.

2.2 Kernel Normalization of Order t

The main purpose of this paper is to introduce a new family of kernel normalization approaches that generalizes the cosine normalization. Our approach amounts to integrating another geometrical based measure that allows us to refine the cosine measure K^0. This additional feature is related to the difference between the norms of the two vectors in the feature space.

These normalization procedures involve generalized mean (also known as power mean) operators which generalize the classical arithmetic mean. Given a sequence of p values $\{a_i\}_{i=1}^p = \{a_1, a_2, \ldots, a_p\}$, the generalized mean with exponent t is given by:

$$\mathcal{M}^t(a_1, \ldots, a_p) = \left[\frac{1}{p}\sum_{i=1}^{p} a_i^t\right]^{\frac{1}{t}} \tag{5}$$

Famous particular cases of (5) are given by $t = -1$, $t \to 0$ and $t = 1$ which are respectively the harmonic, geometric and arithmetic means.

Definition 2 (Kernels normalization of order $t > 0$). *Given a kernel function K, the normalized kernel of order $t > 0$ for any two objects $\mathbf{x}$ and $\mathbf{y}$ of $\mathcal{X}$, is denoted $K^t(\mathbf{x}, \mathbf{y})$ and is defined as follows:*

$$K^t(\mathbf{x}, \mathbf{y}) = \frac{K(\mathbf{x}, \mathbf{y})}{\mathcal{M}^t(K(\mathbf{x}, \mathbf{x}), K(\mathbf{y}, \mathbf{y}))} \tag{6}$$

Similarly to the cosine normalization, $K^t(\mathbf{x}, \mathbf{x}) = 1$ for all $\mathbf{x} \in \mathcal{X}$. As a result, those normalization methods also amount to projecting the objects from the feature space to an unit hypersphere. However, this family of normalized kernels goes beyond the cosine measure since it extends the latter which actually corresponds to the limit case $t \to 0$.

In order to better interpret such measures, let us equivalently formulate $K^t(\mathbf{x}, \mathbf{y})$ with respect to the following norm ratio measures, $\frac{\|\phi(\mathbf{x})\|}{\|\phi(\mathbf{y})\|}$ and $\frac{\|\phi(\mathbf{y})\|}{\|\phi(\mathbf{x})\|}$. We can easily show that:

$$K^t(\mathbf{x}, \mathbf{y}) = \frac{\cos\theta}{\mathcal{M}^t\left(\frac{\|\phi(\mathbf{x})\|}{\|\phi(\mathbf{y})\|}, \frac{\|\phi(\mathbf{y})\|}{\|\phi(\mathbf{x})\|}\right)} \tag{7}$$

This formulation expresses $K^t(\mathbf{x}, \mathbf{y})$ according to geometrical based measures. However, let us introduce the following notation as well:

$$\gamma(\phi(\mathbf{x}), \phi(\mathbf{y})) = \max\left(\frac{\|\phi(\mathbf{x})\|}{\|\phi(\mathbf{y})\|}, \frac{\|\phi(\mathbf{y})\|}{\|\phi(\mathbf{x})\|}\right) = \frac{\max(\|\phi(\mathbf{x})\|, \|\phi(\mathbf{y})\|)}{\min(\|\phi(\mathbf{x})\|, \|\phi(\mathbf{y})\|)} \tag{8}$$

$\gamma(\phi(\mathbf{x}), \phi(\mathbf{y}))$ lies within $[1, +\infty[$ and is related to the difference between the norm measures of $\phi(\mathbf{x})$ and $\phi(\mathbf{y})$. $\gamma(\phi(\mathbf{x}), \phi(\mathbf{y})) = 1$ means that $\|\phi(\mathbf{x})\| = \|\phi(\mathbf{y})\|$ and the greater the difference between the norms' value, the higher $\gamma(\phi(\mathbf{x}), \phi(\mathbf{y}))$.

Similarly to the angular measure, we will denote γ for $\gamma(\phi(\mathbf{x}), \phi(\mathbf{y}))$ in order to simplify the notations. Using γ, we have the different formulations below:

$$K^t(\mathbf{x}, \mathbf{y}) = K^t(\theta, \gamma) = \frac{\cos\theta}{\mathcal{M}^t(\gamma, \gamma^{-1})} = \cos\theta \left(\frac{2^{1/t}\gamma}{(1+\gamma^{2t})^{1/t}} \right) \tag{9}$$

The latter relation expresses $K^t(\mathbf{x}, \mathbf{y})$ as a multiplication between two factors. On the one hand, we have the cosine index $\cos\theta$ and on the other hand, we have the following term which is only dependent on γ, $\left(\frac{2^{1/t}\gamma}{(1+\gamma^{2t})^{1/t}} \right)$.

Following (9), we observe that $\forall\theta : \cos\theta \in [-1, 1]$ and $\forall t > 0, \forall\gamma \geq 1$: $\left(\frac{2^{1/t}\gamma}{(1+\gamma^{2t})^{1/t}} \right) \in]0, 1]$. As a result, one can see that $\forall t > 0, \forall(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^2 : K^t(\mathbf{x}, \mathbf{y}) \in [-1, 1]$.

In what follows, we detail the roles the geometrical parameters θ and γ play with respect to the introduced normalized kernels of order t. First, since $\forall(\phi(\mathbf{x}), \phi(\mathbf{y})) \in \mathcal{F}^2 : \gamma \geq 1$; $K^t(\theta, \gamma)$ is clearly a monotonically increasing function with respect to $\cos\theta$[1]. Then, using (9), we can better underline the effect of the norm ratio γ on $K^t(\theta, \gamma)$. Indeed, by computing the first derivative with respect to this parameter, we obtain:

$$\frac{\partial K^t}{\partial\gamma} = \cos\theta \left[\frac{2^{1/t}(1+\gamma^{2t})^{1/t} \left(1 - \frac{2\gamma^{2t}}{(1+\gamma^{2t})}\right)}{(1+\gamma^{2t})^{2/t}} \right] \tag{10}$$

With the following conditions, $\gamma \geq 1$, $t > 0$, one can verify that in the numerator of the second term of (10), the first factor is positive but the second one, $\left(1 - \frac{2\gamma^{2t}}{(1+\gamma^{2t})}\right)$, is negative since $\frac{2\gamma^{2t}}{(1+\gamma^{2t})} \in [1, 2[$. Consequently, the sign of $\frac{\partial K^t}{\partial\gamma}$ is the same as $-\cos\theta$. Therefore, for $t > 0$, $K^t(\theta, \gamma)$ is monotonically decreasing with respect to γ providing that $\cos\theta > 0$ whereas, $K^t(\theta, \gamma)$ is monotonically increasing with respect to γ as long as $\cos\theta < 0$ (see Fig. 1).

Intuitively, when $t > 0$, the norm ratio measure aims at refining the cosine measure considering that the latter is less and less "reliable" for measuring proximity, as the difference between the vectors' norms becomes larger and larger. Thereby, regardless the sign of the cosine index, the greater γ, the closer to 0 $K^t(\theta, \gamma)$. More formally, we have: $\forall t > 0, \forall\theta, \lim_{\gamma\to+\infty} K^t(\theta, \gamma) = 0$.

Notice that for $t < 0$, we observe the opposite effect since the sign of the derivative $\frac{\partial K^t}{\partial\gamma}$ is the same as $\cos\theta$. Thus, in that case, when $\cos\theta > 0$ for example, $K^t(\theta, \gamma)$ is monotonically increasing with respect to γ. This is not a suitable behavior for a similarity measure, that's the reason why we define $K^t(\theta, \gamma)$ for $t > 0$ only.

In more general terms, the sign and the value of t respectively express the nature and the degree of the influence of γ on $K^t(\theta, \gamma)$. First, when t is negative, it defines a coefficient which is not appropriate for measuring similarity. On

[1] Or a monotonically decreasing function with respect to θ as $\cos\theta$ is a monotonically decreasing function of θ.

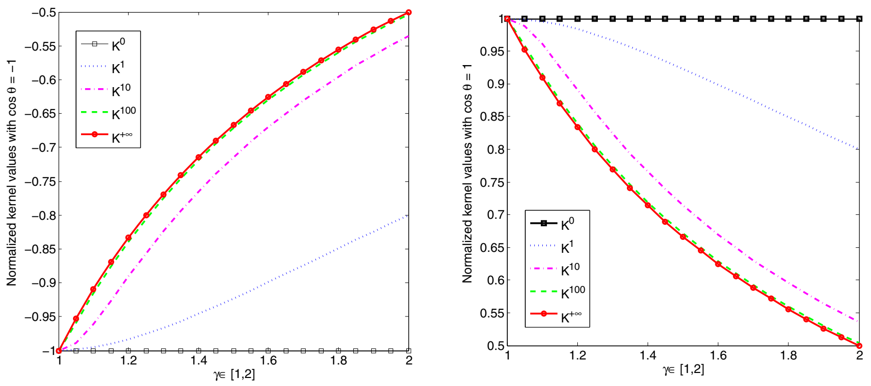


Fig. 1. Curves respectively representing different values of $K^t(-1,\gamma)$ and $K^t(1,\gamma)$ for different t

the contrary, when t is positive it allows one to refine the cosine measure in an appropriate way. Second, assuming that $t > 0$, when the latter increases, it makes γ have a more and more important impact on $K^t(\theta,\gamma)$. In that context, it is worthwhile to mention the two following limit cases: when $t \longrightarrow 0$ we obtain the cosine index which is independent of γ, whereas when $t \longrightarrow +\infty$ we have the following index[2]:

$$K^{+\infty}(\mathbf{x},\mathbf{y}) = \frac{\cos\theta}{\max\left(\frac{\|\phi(\mathbf{x})\|}{\|\phi(\mathbf{y})\|}, \frac{\|\phi(\mathbf{y})\|}{\|\phi(\mathbf{x})\|}\right)} = \frac{\cos\theta}{\gamma}. \tag{11}$$

To illustrate these points, we plotted in Fig. 1, the graphs corresponding to $K^t(\theta,\gamma)$ for different values of t namely the limit when $t \longrightarrow 0$, $t = 1$, $t = 10$, $t = 100$ and the limit when $t \longrightarrow +\infty$. In the left-hand side graph, we fixed $\cos\theta = -1$ and in the right-hand side graph, $\cos\theta$ is set to 1. While the cosine measure is fixed, γ varies from 1 to 2 (the horizontal axis). The goal of these graphs is to represent the effects of the norm ratio parameter γ on the normalized kernel value (the vertical axis) for different values of t and depending on the sign of the cosine measure. For example, when $\cos\theta = 1$ and $\gamma = 2$, we can observe that, in comparison with the case $t \longrightarrow 0$ for which the normalized kernel gives 1, the value drops to 0.85 when $t = 1$ and it drops further to 0.5 when $t \longrightarrow +\infty$.

3 Properties of Normalized Kernels as Similarity Indices

In this section, we want to better characterize the family of normalized kernels that we have introduced. To this end, we give some relevant properties that they respect. First, we show that $K^t(\mathbf{x},\mathbf{y})$ with $t > 0$ satisfies the basic axioms of geometrical based similarity indices. Second, we show that the normalized kernels of order $t > 0$ are kernels. This result is the main theoretical contribution of this paper.

[2] Since we have, $\lim_{t\to+\infty} \mathcal{M}^t(a_1,\ldots,a_p) = \max(a_1,\ldots,a_p)$

3.1 Basic Properties of K^t

We start by giving some basic properties of $K^t(\mathbf{x},\mathbf{y})$ with respect to the general definition of similarity indices defined in a metric space (see for example [6]). $\forall(\mathbf{x},\mathbf{y}) \in \mathcal{X}^2$, we have:

1 $\forall t > 0 : |K^t(\mathbf{x},\mathbf{y})| \leq 1$
2 $\forall t > 0 : K^t(\mathbf{x},\mathbf{x}) = 1$
3 $\forall t > 0 : K^t(\mathbf{x},\mathbf{y}) = K^t(\mathbf{y},\mathbf{x})$
4 $\forall t > 0 : \mathbf{x} = \mathbf{y} \Leftrightarrow K^t(\mathbf{x},\mathbf{y}) = 1$

According to property 1, $K^t(\mathbf{x},\mathbf{y})$ is bounded by -1 and 1 which is a common axiom required for a similarity index. Notice that unnormalized kernels do not respect this axiom in general.

Properties 2 and 3 respectively state that the normalized kernel of order t respects the maximal self-similarity axiom[3] and the symmetric axiom.

According to property 4, the situation $K^t(\mathbf{x},\mathbf{y}) = 1$ corresponds to the case where the vectors are strictly identical in the feature space. Indeed, considering (9), we can see that the normalized kernel value is 1 if and only if $\cos\theta = 1$ and $\gamma = 1$ which is the same as $\phi(\mathbf{x}) = \phi(\mathbf{y})$. Note that property 4 is not true for the limit case $t \to 0$ for which we only have $\phi(\mathbf{x}) = \phi(\mathbf{y}) \Rightarrow K^0(\mathbf{x},\mathbf{y}) = 1$. Since for this case the norm ratio plays no role, it is only sufficient for the vectors to be positively colinear to obtain the maximal similarity value 1. Once again, this shows that the normalized kernels of order $t > 0$ are similarity measures that are more discriminative than the cosine measure. It is also worth mentioning in that context, the two other particular cases: both vectors are completely opposite to each other when we observe $K^t(\mathbf{x},\mathbf{y}) = -1$ ($\cos\theta = -1$ and $\gamma = 1$) and they are geometrically orthogonal when $K^t(\mathbf{x},\mathbf{y}) = 0$ ($\cos\theta = 0$).

In what follows, we focus on the different relationships between normalized kernels of two distinct orders t and t'.

5 $\forall t \geq t' > 0 : \operatorname{sign}(K^t(\mathbf{x},\mathbf{y})) = \operatorname{sign}(K^{t'}(\mathbf{x},\mathbf{y}))$
6 $\forall t \geq t' > 0 : \begin{cases} K^t(\mathbf{x},\mathbf{y}) \leq K^{t'}(\mathbf{x},\mathbf{y}) & \text{if } \cos\theta > 0 \\ K^t(\mathbf{x},\mathbf{y}) \geq K^{t'}(\mathbf{x},\mathbf{y}) & \text{if } \cos\theta < 0 \end{cases}$
7 $\forall t \geq t' > 0 : |K^t(\mathbf{x},\mathbf{y})| \leq |K^{t'}(\mathbf{x},\mathbf{y})|$

Property 5 states that the sign of the similarity measure is independent of t. More precisely, the sign is only dependent on the angular measure. This is a consequence of (9) since $\mathcal{M}^t(\gamma,\gamma^{-1})$ is strictly positive.

Properties 6 and 7 must be put into relation with the comments we made in section 2 and Fig. 1. Accordingly, these properties formally claim that as t grows, $K^t(\mathbf{x},\mathbf{y})$ makes the cosine measure less and less "reliable". We previously mentioned that, all other things being equal, the greater the difference between the vectors' norms in the feature space, the closer to 0 the normalized kernel

[3] Since property 1 states that 1 is the maximal similarity value. Note that this axiom is also called minimality when dealing with dissimilarity rather than similarity [6].

value. These properties express the fact that this convergence is faster and faster as t grows.

These aforementioned properties are direct consequences of the following relations between generalized means, $\forall 0 < t' \leq t < \infty$:

$$\frac{1}{\left(\prod_{i=1}^{p} a_i\right)^{1/p}} > \frac{1}{\mathcal{M}^{t'}(\{a_i\}_{i=1}^{p})} \geq \frac{1}{\mathcal{M}^{t}(\{a_i\}_{i=1}^{p})} > \frac{1}{\max(\{a_i\}_{i=1}^{p})}$$

To complete the analysis of the basic properties that $K^t(\mathbf{x},\mathbf{y})$ respects with regards to the general axioms required for a similarity index, we need to better characterize the metric properties of the latter. We address this issue in the following subsection.

3.2 Metric Properties of K^t

In this paragraph we denote K^t the similarity matrix of objects in $\mathcal{X}$.

Theorem 2. *The similarity matrix K^t with $t > 0$ and general term given by (6) is positive semi-definite. In other words, any normalized kernel K^t with $t > 0$ is a positive semi-definite kernel.*

Proof (Proof of Theorem 2).
The proof of this result is based on Gershgorin circle theorem. Let A be a $(n \times n)$ complex matrix with general term $A_{ii'}$. For each row $i = 1,\ldots,n$, its associated Gershgorin disk denoted $\mathcal{D}_i$ is defined in the complex plane as follows:

$$\mathcal{D}_i = \{z \in \mathbb{C} : |z - A_{ii}| \leq \underbrace{\sum_{i' \neq i} |A_{ii'}|}_{R_i}\} = \mathcal{D}(A_{ii}, R_i)$$

Given this definition, Gershgorin circle theorem (see for example [7]) states that all eigenvalues of A lies within $\bigcup_i \mathcal{D}_i$. Accordingly, if the norms of off-diagonal elements of A are small enough then the eigenvalues are not "far" from the diagonal elements. In other words, the lower the R_i quantities, the closer to the diagonal elements the eigenvalues.

Now, let us consider normalized kernel matrices of order $t > 0$ of objects of $\{\mathbf{x}_i; i = 1,\ldots,n\}$. Let denote $K^t_{ii'} = K^t(\mathbf{x}_i, \mathbf{x}_{i'})$. We first suppose the limit case $t' \to 0$. We know that K^0 is the cosine similarity matrix. As a consequence, K^0 is positive semi-definite and its eigenvalues are all non negative. Next, let us denote $R^t_i = \sum_{i' \neq i} |K^t_{ii'}|$. Then according to properties 2 and 7 given in subsection 3.1, we have:

- $\forall t > 0$ and $\forall i = 1,\ldots,n : K^t_{ii} = 1$,
- $\forall t > t' > 0$ and $\forall i = 1,\ldots,n : R^t_i \leq R^{t'}_i$.

Therefore, when t grows, (6) defines a continuous and differentiable operator for which the absolute value of off-diagonal elements of K^t, and consequently the quantities $R_i^t; i = 1, \dots, n$, are lower and lower while the diagonal entries of K^t remain equal to 1. Thus, applying Gershgorin theorem, we can see that when t increases, the spectrum of K^t is closer and closer to the vector of ones with dimension n. As a consequence, since the eigenvalues of K^0 are non negative then so are the eigenvalues of K^t with $t > t' > 0$ as the latter are closer to 1 than the former. Finally, for $t > 0$, K^t is symmetric and has non negative eigenvalues. These properties are equivalent to the conditions mentioned in Definition 1 thus, for $t > 0$, we can conclude that K^t are positive semi-definite kernels. □

Finally, a corollary of Theorem 2 [8], is that the related distance $D^t(\mathbf{x},\mathbf{y}) = \sqrt{2(1 - K^t(\mathbf{x},\mathbf{y}))}$ respects the triangle inequality axiom, $\forall(\mathbf{x},\mathbf{y},\mathbf{z}) \in \mathcal{X}^3$:

$$\forall t > 0 : D^t(\mathbf{x},\mathbf{y}) \leq D^t(\mathbf{x},\mathbf{z}) + D^t(\mathbf{z},\mathbf{y}) \quad (8)$$

4 Applications to Clustering Tasks

In order to illustrate the potential benefits of our proposal, we tested different normalized kernels of order $t > 0$ in the context of clustering tasks. The clustering algorithm we used is based on kernel Principal Component Analysis (kernel PCA) and the k-means algorithm. Our experiments concern 5 real-world datasets from the UCI ML repository [5]. Our purpose is to show that the normalized kernels that we have introduced, can better capture the proximity relationships between objects compared with other state-of-the-art normalized kernels.

4.1 Kernel PCA Based *k*-means Clustering

Our clustering approach is a spectral clustering like algorithm (see for example [9]). First, from a kernel matrix K^t, we proceed to its eigen-decomposition in order to extract from the implicit high dimensional feature space $\mathcal{F}$, an explicit and proper low dimensional representation of the data. Then, we apply a k-means algorithm in that reduced space in order to find a partition of the objects.

Principal Component Analysis (PCA) is a powerful and widely used statistical technique for dimension reduction and features extraction. This technique was extended to kernels in [3]. Formally, let denote $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_{k-1}$ the leading $k-1$ eigenvalues of K^t and $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{k-1}$ the corresponding eigenvectors. The low dimensional space extracted from K^t that we used as data representation for the clustering step is spanned according to:
$(\sqrt{\lambda_1}\mathbf{v}_1, \sqrt{\lambda_2}\mathbf{v}_2, \dots, \sqrt{\lambda_{k-1}}\mathbf{v}_{k-1})$.

When applying dimension reduction techniques prior to a clustering algorithm, an important issue is the number of dimensions that one has to retain. In this paper, since we aim at using the k-means algorithm as the clustering method, we follow the work presented in [4] that concerns the relationship between k-means and PCA. Accordingly, if k is the number of clusters to find then we retain the $k - 1$ leading eigenvectors.

With regards to related works, we can cite the following papers that use Kernel PCA based clustering in the contexts of image and text analysis respectively, [10,11].

4.2 Experiments Settings

The datasets that we used in our experiments are the following ones [5]:

- Iris (150 objects, 4 features in the input space, 3 clusters)
- Ecoli (336 objects, 7 features in the input space, 8 clusters)
- Pima Indian Diabetes (768 objects, 8 features in the input space, 2 clusters)
- Yeast (1484 objects, 8 features in the input space, 8 clusters)
- Image Segmentation (2310 objects, 18 features in the input space, 7 clusters)

For each data set, we first normalized the data in the input space by centering and standardizing the features. Next, we applied different kernels K namely, the linear kernel $K_l(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$ which is simply the dot product in the input space, and the polynomial kernel $K_p(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + 1)^2$. For each type of kernels, we computed different normalized kernels K^t by varying the value of t: $t \to 0$, $t = 1$, $t = 10$, and the limit $t \to +\infty$. To each of those similarity matrices, we applied the kernel PCA clustering method described previously. Since the k-means algorithm is sensitive with respect to the initialization, for all cases we launched the algorithm 5 times with different random seeds and took the mean average value of the assessment measures.

Since we deal with normalized kernels that amount to projecting the objects on an unit hypersphere, we also tested the kernel PCA based k-means clustering with the Gaussian RBF kernel which presents the same property. This case is our first baseline. However, when using such a kernel, one has to tune a parameter σ. In this paper, to get rid of this problem, we applied the approach proposed in [12] which suggests to use the following affinity measure:

$$K_g(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{\sigma_{\mathbf{x}}\sigma_{\mathbf{y}}}\right) \tag{12}$$

where $\sigma_{\mathbf{x}}$ is set to the value of the distance between $\mathbf{x}$ and its 7th nearest neighbor.

Besides, as our purpose is to show that taking into account the mapped vectors' norm ratio in addition to the cosine measure can be beneficial, we took the case $t \to 0$, which simply corresponds to the cosine index, as a baseline as well.

The assessment measure of the clustering outputs we used is the Normalized Mutual Information (NMI) introduced in [13]. Let denote U the partition found by the kernel PCA clustering and V the ground-truth partition. This evaluation measure is denoted $\mathrm{NMI}(U, V)$ and is given by:

$$\mathrm{NMI}(U, V) = \frac{\sum_{u=1}^{k} \sum_{v=1}^{k} N_{uv} \log\left(\frac{nN_{uv}}{N_{u.}N_{.v}}\right)}{\sqrt{\left(\sum_{u=1}^{k} N_{u.} \log\left(\frac{N_{u.}}{n}\right)\right)\left(\sum_{v=1}^{k} N_{.v} \log\left(\frac{N_{.v}}{n}\right)\right)}} \tag{13}$$

where N is the contingency table between U and V and n the total number of objects. $\mathrm{NMI}(U, V)$ lies in $[0, 1]$ and the higher the measure, the better the clustering output U.

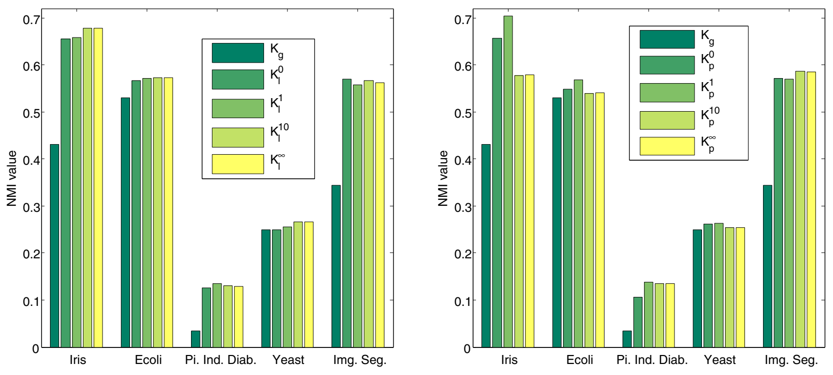


Fig. 2. NMI measures for the different datasets with different kernels (K_l on the left and K_p on the right) and different normalizations ($t \to 0$,$t = 1$,$t = 10$,$t \to +\infty$)

4.3 Experiments Results

In Fig. 2, we report the results we obtained for each dataset. On the left-hand side we give the results related to the linear kernel whereas on the right-hand side, the NMI values correspond to the polynomial kernel. In both graphs, the results provided by the Gaussian RBF kernel are shown (1st bar). Compared to this baseline we can see that the cosine measure and the normalized kernels all perform better on the datasets used in these experiments.

In comparison with the other baseline K^0, we can observe that in most cases there are normalized kernels of order $t > 0$ which can lead to better NMI values. When using the linear kernel, this is true for all K^t except for the Image Segmentation dataset. Furthermore, for the Iris, Ecoli and Yeast datasets, as t grows the performances are consistently better. This shows that taking into account the vectors' norm ratio in the feature space in order to refine the cosine measure, is beneficial.

In the case of polynomial kernel, not all normalization techniques are interesting since many normalized kernels do not outperform the cosine measure. However, when using this kernel, it seems that taking K_p^1 as a normalization method is a good choice. Particularly, K_p^1 leads to the best performances for the Iris and the Pima Indian Diabetes datasets. Besides, concerning the Image Segmentation dataset, we can see that unlike the linear kernel, the normalized polynomial kernels can outperform the cosine index since the best result is obtained with K_p^{10}.

5 Conclusion and Future Work

We have introduced a new family of normalization methods for kernels which extend the cosine normalization. We have detailed the different properties of such methods and the resulting proximity measures with respect to the basic axioms of geometrical based similarity indices. Accordingly, we have shown that normalized kernels are "proper" similarity indices that amount to projecting the data on an unit hypersphere. We have, in addition, proved that these normalized kernels are also kernels. From a practical standpoint, we have also validated the utility of normalized kernels in the context of clustering tasks using several real-world datasets. However, one remaining issue is the choice of the order t when normalizing a kernel. We have shown from a theoretical and a practical point of view that the norm ratio measure can make the normalized kernel more efficient but still, the weight one should give to this parameter in comparison with the angular measure is not straightforward to set. In our future work we intend to further investigate this problem.

References

1. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2001)
2. Vert, J., Tsuda, K., Schlkopf, B.: A primer on kernel methods. In: Schlkopf, B., Vert, J.P., Tsuda, K. (eds.) Kernel Methods in Computational Biology, pp. 35–70. MIT Press, Cambridge (2004)
3. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput. 10(5), 1299–1319 (1998)
4. Ding, C., He, X.: K-means clustering via principal component analysis. In: ICML '04: Proceedings of the twenty-first international conference on Machine learning, p. 29. ACM, New York (2004)
5. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
6. Santini, S., Jain, R.: Similarity measures. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(9), 871–883 (1999)
7. Horn, R., Johnson, C.: Matrix analysis. Cambridge University Press, Cambridge (1985)
8. Gower, J., Legendre, P.: Metric and euclidean properties of dissimilarity coefficients. Journal of classification 3, 5–48 (1986)
9. Filippone, M., Camastra, F., Masulli, F., Rovetta, S.: A survey of kernel and spectral methods for clustering. Pattern Recogn. 41(1), 176–190 (2008)
10. Tuytelaars, T., Lampert, C.H., Blaschko, M.B., Buntine, W.: Unsupervised object discovery: A comparison. International Journal of Computer Vision Epub ahead, 1–19 (July 2009)
11. Minier, Z., Csató, L.: Kernel PCA based clustering for inducing features in text categorization. In: ESANN, pp. 349–354 (2007)
12. Zelnik-Manor, L., Perona, P.: Self-tuning spectral clustering. In: Advances in Neural Information Processing Systems, vol. 17 (2005)
13. Strehl, A., Strehl, E., Ghosh, J.: Cluster ensembles - a knowledge reuse framework for combining partitionings. Journal of Machine Learning Research 3, 583–617 (2002)

Adaptive Matching Based Kernels for Labelled Graphs

Adam Woźnica, Alexandros Kalousis, and Melanie Hilario

University of Geneva, Computer Science Department
7 Route de Drize, Battelle batiment A
1227 Carouge, Switzerland
{Adam.Woznica,Alexandros.Kalousis,Melanie.Hilario}@unige.ch

Abstract. Several kernels over labelled graphs have been proposed in the literature so far. Most of them are based on the Cross Product (CP) Kernel applied on decompositions of graphs into sub-graphs of specific types. This approach has two main limitations: (i) it is difficult to choose a-priori the appropriate type of sub-graphs for a given problem and (ii) all the sub-graphs of a decomposition participate in the computation of the CP kernel even though many of them might be poorly correlated with the class variable. To tackle these problems we propose a class of graph kernels constructed on the proximity space induced by the graph distances. These graph distances address the aforementioned limitations by learning combinations of different types of graph decompositions and by flexible matching the elements of the decompositions. Experiments performed on a number of graph classification problems demonstrate the effectiveness of the proposed approach.

1 Motivation and Related Work

Support Vector Machines (SVMs), and Kernel Methods in general, became popular due to their very good predictive performance [1]. As most of the real-world data can not be easily represented in an attribute-value format many kernels for various kinds of structured data have been proposed [2]. In particular, graphs are a widely used tool for modelling structured data in data mining / machine learning and many kernels over these complex structures have been proposed so far. These kernels have been mainly applied for predicting the activity of chemical molecules represented by sparse, undirected, labelled and two-dimensional graphs.

However, due to the rich expressiveness of graphs it has been proved that kernels over arbitrarily structured graphs, taking their full structure into account, can be neither computed [3] nor even approximated efficiently [4]. The most popular approach to tackle the above problem is based on decompositions of graphs into sub-graphs of specific types which are compared via sub-kernels. The sub-graph types mainly considered are walks [3, 5–8]. However, other researchers have experimented with shortest paths [9], sub-trees [4, 10], cyclic and tree patterns [11] and more general sub-graphs [12, 13].

A common feature in all the above graph kernels is that the *Cross Product (CP) Kernel* is used between the corresponding multi-sets of decompositions. More precisely, for particular decompositions $\mathcal{G}_1^t$ and $\mathcal{G}_2^t$ of the graphs G_1 and G_2 into sub-structures of type t the above kernels can be written as $K(G_1, G_2) = \sum_{(g_1,g_2)\in\mathcal{G}_1^t\times\mathcal{G}_2^t} k(g_1, g_2)$

where k is a kernel over a specific type t of graphs, and the summation over the elements of the multisets takes into account their multiplicity.

One problem with the above kernel is that *all* the possible sub-graphs of a given type are matched by means of a sub-kernel. This might adversely affect the generalization of a large margin classifier since, due to the combinatorial growth of the number of distinct sub-graphs, most of the features in the feature space will be poorly correlated with the target variable [12, 14]. Possible solutions to this problem include down-weighting the contribution of larger sub-graphs [3, 15], using prior knowledge to guide the selection of relevant parts [16], or considering contextual information for limited-size sub-graphs [12]. Yet another solution was proposed in [10] in which only specific elements of the corresponding multi-sets are matched in such a manner that the sum of similarities of the matched elements is maximum. The underlying idea in this kernel is that the actual matching will focus on the most important structural elements, neglecting the sub-structures which are likely to introduce noise to the representation. The idea of using specific pairs of points in a set kernel is promising, however, it is easy to show that this kernel is not positive semi-definite (PSD) in general [17, 18].

More importantly, to the best of our knowledge, all existing graph kernels are currently limited to a *single* type of decomposition which is then, most often, used in the context of the CP kernel. However, in general it is difficult to specify in advance the appropriate type of sub-structures for a given problem. Although, it is in principle possible to simultaneously exploit kernels defined over different representations, this is usually not done because there is a trade-off between the expressivity gained by enlarging the kernel-induced feature space and the increased noise to signal ratio (introduced by irrelevant features). A common intuition is that by decomposing into more complex sub-graphs the expressiveness, and consequently the performance, of resulting kernels increases. This is, however, in contrast with some experimental evidence [12] which suggest that decompositions into rather simple sub-structures perform remarkably well with respect to more complex decompositions on a number of different datasets. A simplified solution to the problem of representation selection is to select the decomposition by cross-validation. However, this approach is problematic since it requires the use of extra data and only one representation can be selected which limits the expressiveness of the resulting method. We can also directly learn to combine graph kernels using multi-kernel learning methods, [19]. Nevertheless, the problem with learning kernel combinations is that the combined elements should, obviously, be valid kernels. However, as we saw previously this type of kernels are based on the CP kernel that requires the complete matching of the components, raising the problems that we described above.

To tackle the above problems we propose a class of adaptive graph kernels built on proximity spaces [20]. The proximity spaces are induced by learned weighted combinations of a given set distance measure on a number of decompositions of different sub-graph types, and constructed on the basis of a representation set, typically the full[1] training set. The weighted combinations are learned from the data, exploiting the NCA method [21] developed for learning Mahalanobis metrics for vectors. By learning the

[1] It is also possible to reduce the size of the representation set relying on feature selection in the proximity space. However, this approach is not discussed in this paper.

weights of the different decompositions we hope to obtain a combination of graph representations that is expressive enough for the task at hand and does not overfit. Using the set distance measures on the graph decompositions allows for flexible ways of mapping the elements of these decompositions, namely it is not necessary to use all the elements (sub-graphs) in the mapping. We originally explored the use of proximity spaces to define a graph kernel in [22]; however, there we were limited to a fixed decomposition and did not consider learning the combinations of the different types of substructures.

In this paper we will experiment with, and learn combinations of, two types of decompositions: walks and unordered trees of various lengths and heights, respectively. The former were shown to be very effective in chemical domains and achieved state-off-the-art results [3, 5–8]. Decomposition kernels based on trees were first proposed in [4], however, experimental evaluation was not performed. Kernels based on trees were examined in [10]. Walks and trees can be easily compared by a graded similarity (e.g. standard Euclidean metric for walks). Nevertheless, most of the kernels on graphs use the Kronecker Delta Kernel (i.e. $k_\delta(x, y) = 1$ if $x = y$, $k_\delta(x, y) = 0$ otherswise) on sub-parts. As a result, the ability to find partial similarities is lost and the expressivity of these kernels is reduced [7]. A graded similarity on walks was considered in the graph kernel presented in [7], however, it suffers from high computational complexity since it requires taking powers of the adjacency matrix of the direct product graph, leading to huge runtime and memory requirements [9]. Graded similarities on trees were also used in [10]. Finally, we note that our framework is not limited only to walks and trees and can be applied on decompositions of any type.

This paper is organized as follows. In Sect. 2 we define all the necessary notions of graphs. In Sect. 3 we define the adaptive kernels on graphs in the proximity space induced by the set distance measures. Experimental results are reported in Sect. 4. Finally, Sect. 5 concludes the work.

2 Primer of Graph Theory

An *undirected* graph $G = (\mathcal{V}, \mathcal{E})$ is described by a finite set of *vertices* $\mathcal{V} = \{v_1, \ldots, v_k\}$ and a finite set of *edges* $\mathcal{E}$, where $\mathcal{E} = \{\{v_i, v_j\} : v_i, v_j \in \mathcal{V}\}$; for *directed* graphs we set $\mathcal{E} = \{(v_i, v_j) : v_i, v_j \in \mathcal{V}\}$. We shall also use the following notation for edges: $e_{ij} = \{v_i, v_j\}$ (or $e_{ij} = (v_i, v_j)$). For *labelled* graphs there is additionally a set of vertex labels $\mathcal{L}_\mathcal{V}$ and edge labels $\mathcal{L}_\mathcal{E}$ together with functions $l_\mathcal{V} : \mathcal{V} \to \mathcal{L}_\mathcal{V}$ and $l_\mathcal{E} : \mathcal{E} \to \mathcal{L}_\mathcal{E}$ that assign labels to vertices and edges, respectively. We will use $lab(x)$ to denote, in a more general form the label of x; whether $lab(x) = l_\mathcal{V}(x)$ or $lab(x) = l_\mathcal{E}(x)$ will be clear from the type of the x argument, i.e. vertex or edge. In this work we assume that the labels are vectors, i.e. $\mathcal{L}_\mathcal{V} \subseteq \mathbb{R}^p$ and $\mathcal{L}_\mathcal{E} \subseteq \mathbb{R}^n$ for some values of m and n. By $dim(lab(x))$ we will denote the dimensionality of the label vector $lab(x)$ which obviously will be the dimensionality of either $\mathcal{L}_\mathcal{V}$ or $\mathcal{L}_\mathcal{E}$, depending again on x.

Some special types of graphs that are relevant to our work are walks and trees. A *walk*, W, in a graph G is a sequence of vertices and edges, $W = [v_1, e_{12}, v_2, \ldots, e_{s,s+1}, v_{s+1}]$, such that $v_j \in \mathcal{V}$ for $1 \leq j \leq s+1$ and $e_{ij} \in \mathcal{E}$ for $1 \leq i \leq s$. Walks can also end to an edge, e.g. $W = [v_1, e_{12}, v_2, \ldots, e_{s,s+1}]$. The length l of a walk, W, is the number of vertices and edges in W. We denote the set of all walks of length l in a graph G by $\mathcal{W}(G)^l$. A *tree* T with vertices $\mathcal{V}$ and edges $\mathcal{E}$ of a graph G is a sub-graph of G which is connected and acyclic. The root node of a tree is denoted as $root(T)$. In this work we will only consider directed trees where the order is given from the root node to the leafs. The height h of a tree T is the length of the longest walk from the root node to any of the leaf nodes (similar to walks we allow that the trees have edges as leafs). We denote the set of all directed trees of height h in a graph G by $\mathcal{T}(G)^h$. We define the neighborhood of a node, v, in a tree as $\delta(v) = \{e : e = (v,u) \in \mathcal{E}\}$, and the neighborhood of an edge, e, as $\delta(e) = \{u : e = (v,u) \in \mathcal{E}\}$, note that in fact the neighborhood of an edge is a one element set, containing a single node.

3 Kernels on Graphs

We denote the decomposition of a graph, G, into the *multi-set* of sub-graphs of type t by $\mathcal{G}^t$. We focus on decompositions into walks of various lengths l, i.e. $\mathcal{G}^t = \mathcal{W}(G)^l$, and (directed) trees of various heights h, i.e. $\mathcal{G}^t = \mathcal{T}(G)^h$. More generally, a graph G can be represented by a tuple of m different decompositions as $\boldsymbol{\mathcal{G}} = (\mathcal{G}^{t_1}, \ldots, \mathcal{G}^{t_m})^T$.

In this section we will define a class of adaptive kernels for labelled graphs. The construction of these kernels is based on set distance measures which will be used to compute the distances on the components of $\boldsymbol{\mathcal{G}}$. Since it is difficult to select a priori the appropriate types of sub-structures, i.e. components of $\boldsymbol{\mathcal{G}}$, for a given problem, we propose a method which learns a weighted (quadratic) combination of a fixed set distance on the different components of $\boldsymbol{\mathcal{G}}$. Finally, we use the learned weighted combination of the set distance to induce a proximity space on which we define the final kernel.

Distances on Sets. The central idea in the set distance measures that we consider is the definition of a mapping of the elements of one set to the elements of the other such that the final distance is determined on the basis of specific pairs of elements from the two sets. More precisely, consider two nonempty and finite sets $A = \{a\} \subseteq \mathcal{X}$ and $B = \{b\} \subseteq \mathcal{X}$. Let $d(\cdot,\cdot)$ be a distance measure defined on $\mathcal{X}$. The set distance measure d_{set} is defined on $2^{\mathcal{X}}$ as $d_{\mathrm{set}}(A,B) = \frac{\sum_{(a,b)\in F} d(a,b)}{|F|}$ i.e. it is a normalized sum of pairwise distances over specific pairs which are defined by $F \subseteq A \times B$; different F correspond to different set distance measures. Within this framework we can define *Average Linkage (d_{AL})*, *Single Linkage (d_{SL})*, *Complete Linkage (d_{CL})*, *Sum of Minimum Distances (d_{SMD})*, *Hausdorff (d_{H})*, *RIBL (d_{RIBL})*, *Tanimoto (d_{T})*, *Surjections (d_{S})*, *Fair Surjections (d_{FS})*, *Linkings (d_{L})* and *Matchings (d_{M})* distance measures. Detailed description of these distance measures can be found in [23].

Distances Between Sets of Decompositions. We will now show how to use the above set distance measures to compute distances between sets of decompositions of graphs to sub-graphs of specific types. As already mentioned, we will focus on decompositions into walks and (directed) trees of various lengths and heights. In the latter, we consider *unordered trees* where the order among the siblings at a given height is not important. Both types of decompositions are obtained from a depth first exploration emanating from each node in a graph. We note that the walks and trees we consider in this work are non-standard in the sense that a walk can end in an edge and a tree can have edges as leafs.

In our decompositions we do not allow repetitions of a node within a walk or a tree, i.e. we do not allow cycles, in order to avoid the problem of *tottering* [6]. We exclude from the graph decomposition walks of the form $W = [v_1, e_{12}, v_2, \ldots, e_{s,s+1}, v_{s+1}]$ with $v_i = v_{i+2}$ for some i since these are likely to introduce redundancy. In the case of trees we do not allow such walks in the trees' descriptions. Even though tottering-free graph representations do not seem to bring an improvement to the predictive performance [6, 10], they have the advantage of inducing decompositions smaller in size. This has a direct influence on the computational complexity of the set distance measures applied over this representation.

The main difficulty in applying the set distance measures from the previous section is to define the distance, d, between the elements of the sets, i.e. walks or trees. To define a distance measure over walks we exploit the (normalized) Euclidean metric. More precisely, the distance of two walks W_i and W_j of equal length l is defined as the Euclidean metric between the labels of the elements of the walks $d^2_W(W_i, W_j) = \frac{\sum_{k=1}^{l} d^2_{\mathrm{lab}}(lab(W_i[k]), lab(W_j[k]))}{N}$ where $W[k]$ is the k-th element of walk W, and $d_{\mathrm{lab}}(\cdot,\cdot)$ is the Euclidean metric between the labels of corresponding walks. Obviously, $W_i[k]$ and $W_j[k]$ are of the same type and they will be either edges or vertices. The N in the denominator is a normalization factor and corresponds to the sum of the dimensionalities of the label vectors of the l elements that make up the paths, obviously $N = \sum_{k=1}^{l} dim(lab(W_i[k]) = \sum_{k=1}^{l} dim(lab(W_j[k])$. We have $0 \leq d^2_{\mathrm{W}}(\cdot,\cdot) \leq 1$.

Lets now define a distance, $d_{\mathrm{T}}(T_i, T_j)$, between two trees, T_i, T_j; we should note here that unlike walks, trees do not have to be of the same height. Let x, y, be two elements of the two trees found at the same height h. These elements will be either two vertices, u, v, or two edges e_i, e_j. Then the (squared) distance between x and y, $d^2_{\mathrm{t}}(x, y)$, is recursively defined as:

$$\begin{cases} \frac{d^2_{\mathrm{lab}}(lab(x),lab(y)) + d^2_{\mathrm{M}}(\delta(x),\delta(y))}{N'} & \text{if } \delta(x) \neq \emptyset \wedge \delta(y) \neq \emptyset \\ \frac{d^2_{\mathrm{lab}}(lab(x),lab(y))}{N'} & \text{if } \delta(x) = \emptyset \wedge \delta(y) = \emptyset \\ \frac{d^2_{\mathrm{lab}}(lab(x),lab(y))+1}{N'} & \text{if } \delta(x) = \emptyset \otimes \delta(y) = \emptyset \end{cases}$$

where $\otimes$ is the logical XOR; $d_{\mathrm{lab}}(lab(x), lab(y))$ is the Euclidean metric between the labels of the x, y; $\delta(x)$ is the neighbourhood function, defined in Sect. 2, that returns either: the set of edges to which a vertex connects to as a starting vertex, if x is of type vertex, or the vertex to which an edge arrives if x is of type edge; finally d_{M} is the matching set distance measure between the sets of elements that are found in the neighborhoods of x and y. The N' in the denominator is also a normalization factor that corresponds to the dimensionality of the label vectors of x and y plus one to account for the set distance dimension, i.e. $N' = dim(lab(x)) + 1 = dim(lab(y)) + 1$. The matching distance d_{M} will establish the best mapping among the elements of $\delta(x), \delta(y)$, letting outside, and penalizing for, sub-structures that cannot be matched. In a discrete case scenario it is equivalent to a frequency based distance, i.e. for each of the discrete sub-structures in $\delta(x), \delta(y)$, it will count how many times it appears and the final distance will be the sum of differences of these frequencies. Note here that $d_{\mathrm{t}}(x, y)$ is a recursive distance requiring the computation of set distances between the elements of the trees that are associated to x, y, at the $h+1$ height. The final distance between two trees, T_i, T_j, is given by $d_{\mathrm{T}}(T_i, T_j) = d_{\mathrm{t}}(root(T_i), root(T_j))$.

Combining Different Decompositions. Here we show how to combine different decompositions of labelled graphs in order to produce a final weighted graph distance. Recall that two graphs G_1 and G_2 can be represented by m different decompositions as $(\mathcal{G}_1^{t_1}, \ldots, \mathcal{G}_1^{t_m})^T$ and $(\mathcal{G}_2^{t_1}, \ldots, \mathcal{G}_2^{t_m})^T$ of types $t = t_1, \ldots, t_m$. For a fixed set distance measure d_{set} and the given m decompositions we define the vector $\boldsymbol{d}(G_1, G_2) = (d_{\mathrm{set}}(\mathcal{G}_1^{t_1}, \mathcal{G}_2^{t_1}), \ldots,\ d_{\mathrm{set}}(\mathcal{G}_1^{t_m},\ \mathcal{G}_2^{t_m}))^T$ which consists of the distances over the different decompositions. Then the weighted combination of these decompositions is defined as

$$d^2_{\boldsymbol{A}}(G_1, G_2) = \boldsymbol{d}(G_1, G_2)^T \boldsymbol{A}^T \boldsymbol{A}\ \boldsymbol{d}(G_1, G_2) \tag{1}$$

where $\boldsymbol{A}$ is a $m \times m$ matrix. It should be noted that for any matrix $\boldsymbol{A}$ the matrix $\boldsymbol{A}^T\boldsymbol{A}$ is PSD, and hence $d_{\boldsymbol{A}}$ is a pseudo-metric. Here we propose a method for learning the matrix $\boldsymbol{A}$ of equation (1) directly from the data by casting the problem as an optimization task.

To learn the matrix $\boldsymbol{A}$ we use the Neighborhood Component Analysis (NCA) method from [21] which was originally developed for metric learning over vectorial data.[2] This method attempts to directly optimize a continuous version of the leave-one-out (LOO) error of the kNN algorithm on the training data. First, a conditional distribution is introduced which for each example G_i selects another example G_j as its neighbor with some probability $p_{\boldsymbol{A}}(j|i)$, and inherits its class label from the point it selects. The probability $p_{\boldsymbol{A}}(j|i)$ is based on the softmax of the $d^2_{\boldsymbol{A}}$ distance given by $p_{\boldsymbol{A}}(j|i) = \frac{e^{-d^2_{\boldsymbol{A}}(G_i, G_j)}}{\sum_{k \neq i} e^{-d^2_{\boldsymbol{A}}(G_i, G_k)}}$, $p(i|i) = 0$. We denote C_i as the set of points that share the same class with G_i, i.e. $C_i = \{j \mid class(G_i) = class(G_j)\}$. The objective function to be maximized, as described in [21], is $\mathcal{F}_{\boldsymbol{A}} = \sum_i \log(\sum_{j \in C_i} p_{\boldsymbol{A}}(j|i))$.

It is clear that for full matrices $\boldsymbol{A}$ the number of parameters to estimate is m^2. This could be problematic in cases where m is large with respect to the number of instances in the training database. One possible solution to overcome this problem is to add a soft constraint to the above objective function which results in the following regularized objective function $\mathcal{F}^{\mathrm{reg}}_{\boldsymbol{A}} = \sum_i \log(\sum_{j \in C_i} p_{\boldsymbol{A}}(j|i)) + \lambda \|\boldsymbol{A}\|^2_{\mathcal{F}}$ where $\|\boldsymbol{A}\|_{\mathcal{F}}$ denotes a Frobenious norm of matrix $\boldsymbol{A}$ and $\lambda > 0$ is a regularization parameter. The other solution is to restrict matrix $\boldsymbol{A}$ to be diagonal resulting in a weighted combination of distances for different decompositions (we denote NCA where optimization is performed over a diagonal matrix by $\mathrm{NCA}_{\mathrm{diag}}$). The main advantage of the NCA algorithm is that it is non-parametric, making no assumptions about the shape of the class conditional distributions [21]. Its main problem is that the objective function is not convex, hence some care should be taken to avoid local optima.

Adaptive Matching Based Kernels. Here we exploit the adaptive combinations of decompositions presented in Sect. 3, and the notion of proximity spaces, to define a class of adaptive matching based kernels over labelled graphs. The proximity space is

[2] We also experimented with other metric learning methods; however, due to the lack of space, these results are not reported in this paper.

defined by an instantiation of $d_{\boldsymbol{A}}$ from equation (1) and a representation set (i.e. set of prototypes) of learning instances. More precisely, consider a graph G which is decomposed in m different ways as $\boldsymbol{\mathcal{G}} = (\mathcal{G}^{t_1}, \ldots, \mathcal{G}^{t_m})^T$. For a given representation set $S = \{G_1, \ldots, G_p\}$ (we assume that each graph is uniquely identified by its index) and a distance measure $d_{\boldsymbol{A}}$ we define a mapping $\boldsymbol{d_A}(G, S) = (d_{\boldsymbol{A}}(G, G_1), \ldots, d_{\boldsymbol{A}}(G, G_p))^T$. Since distance measures $d_{\boldsymbol{A}}$ are non-negative, all the data examples are projected as points to a non-negative orthotope of that vector space. The dimensionality of this space is controlled by the size of the set S. In this setting the adaptive graph kernel in the proximity space between graphs G_1 and G_2 is defined as $k_{d_{\boldsymbol{A}}}(G_1, G_2) = k(\boldsymbol{d_A}(G_1, S), \boldsymbol{d_A}(G_2, S))$, where k denotes an elementary kernel in the proximity space; k can be any standard kernel on vectorial data.

We mention that the complexity of computing the adaptive matching kernel between two graphs (if the weights of different decompositions are known) can be shown to be at most $O\{|S|m^2|G|^3(\alpha^{3l} + \alpha^{3h})\}$ where $|S|$ is the cardinality of the representation set, $|G|$ is the number of vertices in graph G, α is the branching factor which can be upper bounded by a small constant (usually 4), and $m = l + h$ is the number of decompositions. The complexity of learning the weight is at most $O\{m^2(n^2|G|^3 + n^3)\}$ where n is the number of graphs in the training set.

4 Experiments

We will experiment with two graph classification problems: Mutagenesis and Carcinogenicity. The application task in the Mutagenesis dataset is the prediction of mutagenicity of a set of 188 aromatic and hetero aromatic nitro-compounds which constitute the "regression friendly" version of this dataset. The other classification problem comes from the Predictive Toxicology Challenge and is defined over carcinogenicity properties of chemical compounds. This dataset lists the bioassays of 417 chemical compounds for four type of rodents: male rats (MR), male mice (MM), female rats (FR) and female mice (FM) which give rise to four independent classification problems. We transformed the original dataset (with 8 classes) into a binary problem by ignoring EE (equivocal evidence), E (equivocal) and IS (inadequate study) classes, grouping SE (some evidence), CE (clear evidence) and P (positive) in the positive class and N (negative) and NE (no evidence) in the negative one. After this transformation the MR, MM, FR and FM datasets had 344, 336, 351 and 349 instances, respectively. References to these datasets can be found in [22].

In the experiments we want to examine a number of issues related to our graph kernels. More precisely, first, for different set distance measures we will explore the effect of the length of walks (and the height of trees) on the performance of $k_{d_{\text{set}}}$, i.e. we fix the walk length to a specific l for $l = 1, \ldots, 11$ (and the height of a tree is fixed to a specific h, $h = 2, \ldots, 5$) and we use only the corresponding decomposition to construct the final kernel; we do *not* combine decompositions. We will compare the performance of the above simplified kernels with $k_{d_{\boldsymbol{A}}}$ where we combine all the different $m = 11 + 4$ decompositions. As the kernel k on the proximity space we will be always using the linear kernel in order to make a fair comparison between the algorithms and to avoid the situation where an implicit mapping given by a nonlinear kernel will influence the

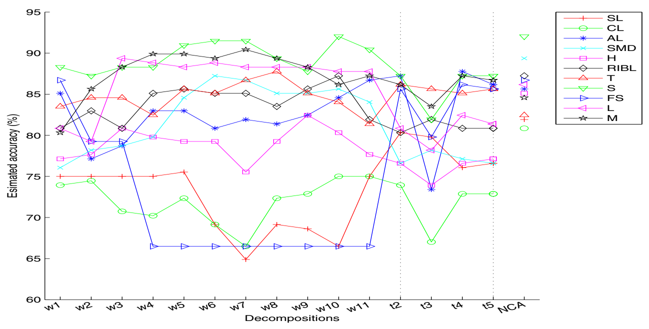


Fig. 1. Estimated accuracy of different kernels in the proximity space ($k_{d_{\text{set}}}$) vs. different decompositions for different set distance measures in the Mutagenesis dataset. wl denotes walks of length l whereas th denotes trees of height h. The last values in the plot denote the performance of $k_{d_{\boldsymbol{A}}}$ where the different decompositions are combined.

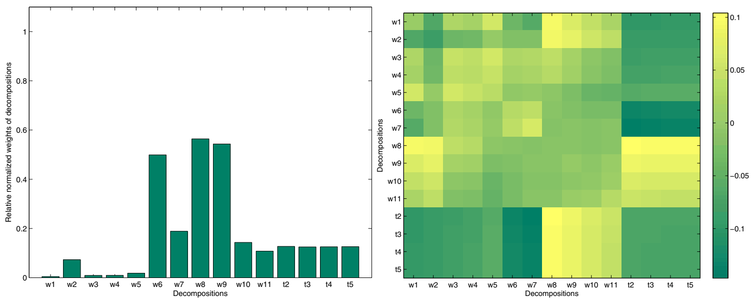
results. Second, we will examine how the performance of the kernel $k_{d_{\boldsymbol{A}}}$ compares with the following two set kernels based on averaging: (i) the direct sum kernel [1] based of the cross product kernels applied on the different $m = 11 + 4$ decompositions with the linear kernel (in case of walks) and the tree kernel[3] (for trees) as kernels on the sets' elements ($k_{\Sigma,\text{CP}}$), and (ii) the linear kernel in the proximity space induced by $d_{\boldsymbol{A}}$, also over the combination of the different $m = 11 + 4$ decompositions, where d_{AL} set distance measure is applied over the decompositions. Finally, we will compare our graph kernels with other graph kernels form the literature.

We use the SVM algorithm where the parameter C is optimized in an inner 10-fold cross-validation loop over the set $C = \{0.1, 1, 10, 50\}$. In all the experiments, unless stated otherwise, we use the regularized version of the NCA algorithm over full matrices $\boldsymbol{A}$, where the regularization parameter λ is internally 10-fold cross-validated over $\lambda = \{0, 0.1, 1, 10\}$. In some experiments we also use the version of NCA, denoted as NCA_{diag}, where $\boldsymbol{A}$ is limited to a diagonal matrix. Note that in the experiments we have 11 different instantiations of the $k_{d_{\boldsymbol{A}}}$ kernel, each one corresponding to one of the 11 set distance measures. In all the experiments accuracy is estimated using stratified 10-fold cross-validation and controlled for the statistical significance of observed differences using McNemar's test, with significance level of 0.05.

4.1 Results and Analysis

Performance of Individual vs. Combined Decompositions. We first examine the influence of the length of the walks and heights of the trees to the predictive performance of SVM. The results for the Mutagenesis dataset are presented in Fig. 1. From the results it is clear that in Mutagenesis the optimal decomposition depends on the actual set distance measure. For example, for d_{SMD} the highest predictive accuracy is obtained

[3] The definition of the tree kernel is based on recursive and alternating computations of the linear and cross product kernels [1]; the way these kernels are combined to compute the final tree kernel is similar to the definition of the tree distance from Sect. 3.



(a) $\mathrm{NCA}_{\mathrm{diag}}$ (b) NCA, $\lambda = 0$

Fig. 2. Relative importance of different decompositions (FM dataset) for the d_{SMD} set distance measure both for diagonal and full matrices $\boldsymbol{A}$. wl denotes walks of length l whereas th denotes trees of height h. It is represented as normalized weights $\boldsymbol{A}$.

for walks of length 6, whereas for d_{S} the best decomposition is into walks of length 10. Additionally, with the exception of d_{SL}, decompositions in walks in general outperform decompositions into trees. For the other examined datasets the optimal decompositions are different. Thus we have no way to know a-priori which type of decomposition is appropriate. By combining them using NCA (results also given in Fig. 1) we get a classification performance that is in most cases as good as that of the single decompositions. Indeed, in Mutagenesis the performance of NCA was significantly better in 68 cases, in 92 cases the differences were not significantly meaningful, and in 5 cases it was significantly worse.[4] In FM the corresponding values are 7, 158 and 0; in FR: 5, 147 and 22; in MM: 12, 153 and 0; in MR: 81, 83 and 1. We should stress here that it is not fair to compare NCA with the results of the best decomposition, since this estimate is optimistically biased since we need to have the results of the cross-validation to determine which is the best decomposition. In a fair comparison NCA should be compared to a model selection strategy in which the best decomposition is selected using an inner cross-validation. However an inner cross-validation-based model selection strategy does not make full use of the data, thus its estimates might not be reliable, and it is computationally expensive.

The results of the optimization process that learns the optimal combination of decompositions can be graphically presented providing insight to the relative importance of different decompositions. In Fig. 2 we give an example of such a visualization for the FM dataset and the d_{SMD} set distance measure. In the left graph of that figure the optimization was performed for diagonal matrices using $\mathrm{NCA}_{\mathrm{diag}}$. The different elements in x-axis are the elements of the diagonal of the matrix $\boldsymbol{A}$ which correspond to a decomposition into walks and trees whereas the y-axis represents the weights, normalized by a Frobenious norm of $\boldsymbol{A}$, returned by the optimization method. What we see from the graph is that in FM the highest weights are assigned to walks of lengths longer than 6 and all tress, independent of their heights. The right graph of Fig. 2 provides the visualization of the optimization process for NCA, where now the matrix $\boldsymbol{A}$ is a full

[4] The total number of comparisons is 165 = 11 walk decompositions x 11 set distances + 4 tree decompositions x 11 set distances.

Table 1. Accuracy and significance test results of SVM in the graph datasets (the + sign stands for a significant win of the first algorithm in the pair, - for a significant loss and = for no significant difference). The sign in the first parenthesis corresponds to the comparison of SVM vs. SVM with $k_{d_{\boldsymbol{A}}}$ with d_{AL}, while the sign in the second parenthesis compares SVM vs. SVM with $k_{\Sigma,\mathrm{CP}}$.

Set Distance	Mutagenesis	FM	FR	MM	MR
d_{SL}	81.91 (=)(=)	65.04 (=)(=)	67.23 (=)(=)	66.96 (=)(=)	63.95 (=)(+)
d_{CL}	80.85 (=)(=)	60.46 (=)(=)	63.24 (-)(=)	65.18 (=)(=)	67.73 (+)(+)
d_{SMD}	89.36 (=)(+)	62.17 (=)(=)	67.52 (=)(=)	66.37 (=)(=)	65.99 (=)(+)
d_{H}	85.11 (=)(=)	64.18 (=)(=)	66.09 (=)(=)	66.07 (=)(=)	65.11 (=)(+)
d_{RIBL}	87.23 (=)(=)	64.76 (=)(=)	67.52 (=)(=)	66.66 (=)(=)	67.15 (+)(+)
d_{T}	82.45 (=)(=)	64.76 (=)(=)	67.81 (=)(=)	68.45 (=)(+)	68.02 (+)(+)
d_{S}	92.02 (+)(+)	64.18 (=)(=)	67.52 (=)(=)	66.96 (=)(=)	61.34 (=)(=)
d_{FS}	86.70 (=)(=)	61.60 (=)(=)	64.10 (=)(=)	63.99 (=)(=)	61.34 (=)(=)
d_{L}	86.17 (=)(=)	64.18 (=)(=)	65.81 (=)(=)	64.88 (=)(=)	66.28 (+)(+)
d_{M}	84.57 (=)(=)	64.46 (=)(=)	66.66 (=)(=)	65.48 (=)(=)	67.73 (+)(+)
d_{AL}	85.64	62.46	66.38	66.07	60.46
$k_{\Sigma,\mathrm{CP}}$	84.04	62.46	64.96	63.39	57.85

matrix ($\lambda = 0$). One surprising observation is that for NCA the decompositions into trees and long walks are assigned low weights. At the same time combinations of trees with long walks and combinations of shorter walks are of high importance.

Performance of Matchings Strategies. The next dimension of comparison is the relative performance of the instantiations of the $k_{d_{\boldsymbol{A}}}$ kernel for different set distances and the following two set kernels based on averaging: (i) the direct sum kernel [1] based on the cross product kernels applied on the different 15 decompositions ($k_{\Sigma,\mathrm{CP}}$), and (ii) the linear kernel in the proximity space induced by $d_{\boldsymbol{A}}$, also over the combination of the different 15 decompositions, with the d_{AL} set distance measure.

The results are presented in Table 1. The relative performance of kernels based on specific pairs of elements and kernels based on averaging depends on the actual application. For Mutagenesis and MR there is an advantage of the kernels based on specific pairs of elements, while for the remaining datasets the performances are similar. Overall, the choice of the appropriate way of matching the elements of two sets depends on the application and ideally should be guided by domain knowledge, if such exists. Nevertheless, the relative performance of the different kernels provides valuable information about the type of problem we are facing. For example, by examining Mutagenesis and MR we see that averaging performs poorly, indicating that the local structure of the molecules is important, while in the remaining datasets both global and local structures seem to be equally informative.

Comparison with Other Graph Kernels. In Table 2 we provide the best results reported in the literature on the same benchmark datasets.[5] Additionally, we report the performance of kernels corresponding to d_{SMD} and d_{T}, both achieving stable good

[5] We only cite works which use similar features to describe atoms and bonds. In particular our results for Carcinogenicity are not directly comparable with the ones reported in [10].

Table 2. Comparison with other related graph kernels

Graph kernels	Mutagenesis	FM	FR	MR	MM
Best kernel from [5]	85.1	63.4	66.1	58.4	64.3
Best kernel from [8]	91.5	64.5	66.9	65.7	66.4
Based on d_{SMD}	89.4	62.2	67.5	66.4	66.0
Based on d_{T}	82.4	64.8	67.8	68.4	68.0
Our best kernel	92.0	65.0	67.8	68.5	68.0

results across the different datasets. The values in the ”Our best kernel” row correspond to the best kernel for each dataset, selected over all the examined mappings. It is obvious that the latter results are optimistically biased since they are selected after extensive experimentation with various set distance measures. However, the same could be argued for the results of all the other related kernels given in Table 2 since in all the cases multiple results where available and we reported on the best. The corresponding results show that if the mapping is carefully selected for a problem at hand, then the performance of the corresponding kernel is better than the performance of the existing graph kernels. At the same time kernels corresponding to d_{SMD} and d_{T} achieve stable results which are close to the state-of-the-art for the considered datasets.

5 Conclusions and Future Work

It has been argued in [7] that a “good” kernel for graphs should fulfil at least the following requirements: (i) should be a good similarity measure for graphs, (ii) its computational time should be possible in polynomial time, (iii) should be positive semi-definite and (iv) should be applicable for various graphs. In this paper we proposed a class of adaptive kernels for graphs which are based on walks and trees, that are computable in polynomial time, that are PSD, and are applicable to a wide range of graphs. A distinctive feature of our kernels is that they allow for (i) combinations of different types of decompositions and (ii) specific types of mappings between sub-parts. The effectiveness of the approach was demonstrated on problems of activity prediction of chemical molecules. Finally, the ideas presented in this work are not limited to graphs and can be directly exploited to define kernels over not only special kinds of graphs like sequences and trees but also over instances described using first- (or higher-)order logic [24].

In the future work we would like to examine decompositions of graphs into substructures other than walks and trees (e.g. the cyclic patterns from [11]). Moreover, we will apply the methods developed in this work on larger datasets, e.g. the HIV dataset described in [12].

Acknowledgments. This work was partially funded by the European Commission through EU projects DebugIT (FP7-217139) and e-LICO (FP7-231519). The support of the Swiss NSF (Grant 200021-122283/1) is also gratefully acknowledged.

References

1. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
2. Gärtner, T.: A survey of kernels for structured data. SIGKDD Explor. Newsl. 5(1), 49–58 (2003)
3. Gärtner, T., Flach, P., Wrobel, S.: On graph kernels: Hardness results and efficient alternatives. In: Proceedings of COLT 16 and the 7th Kernel Workshop (2003)
4. Ramon, J., Gärtner, T.: Expressivity versus efficiency of graph kernels. In: First International Workshop on Mining Graphs, Trees and Sequences (held with ECML/PKDD'03) (2003)
5. Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized kernels between labeled graphs. In: ICML, Washington, DC (2003)
6. Mahé, P., Ueda, N., Akutsu, T., Perret, J.L., Vert, J.P.: Extensions of marginalized graph kernels. In: ICML (2004)
7. Borgwardt, K.M., Ong, C.S., Schönauer, S., Vishwanathan, S., Smola, A.J., Kriegel, H.P.: Protein function prediction via graph kernels. Bioinformatics 21(1), i47–i56 (2005)
8. Ralaivola, L., Swamidass, S.J., Saigo, H., Baldi, P.: Graph kernels for chemical informatics. Neural Networks, 1093–1110 (2005)
9. Borgwardt, K.M., Kriegel, H.P.: Shortest-path kernels on graphs. In: ICDM (2005)
10. Fröhich, H., Wegner, J., Sieker, F., Zell, A.: Optimal assignment kernels for attributed molecular graphs. In: ICML (2005)
11. Horváth, T., Gärtner, T., Wrobel, S.: Cyclic pattern kernels for predictive graph mining. In: KDD (2004)
12. Menchetti, S., Costa, F., Frasconi, P.: Weighted decomposition kernels. In: ICML (2005)
13. Sherashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., Borgwardt, K.: Efficient graphlet kernels for large graph comparison. In: 12th AISTATS (2009)
14. Ben-David, S., Eiron, N., Simon, H.: Limitations of learning via embeddings in euclidean half spaces. Journal of Machine Learning Research 3, 441–461 (2002)
15. Collins, M., Duffy, N.: Convolution kernels for natural language. In: NIPS (2002)
16. Cumby, C., Roth, D.: On Kernel Methods for Relational Learning. In: ICML (2003)
17. Woźnica, A., Kalousis, A., Hilario, M.: Distances and (indefinite) kernels for sets of objects. In: ICDM (2006)
18. Vert, J.P.: The optimal assignment kernel is not positive definite (2008)
19. Lanckriet, G., Cristianini, N., Bartlett, P.L., Ghaoui, L.E., Jordan, M.: Learning the kernel matrix with semi-definite programming. Journal of Machine Learning Research 5 (2004)
20. Pekalska, E., Duin, R.: The Dissimilarity Representation for Pattern Recognition. In: Foundations and Applications. World Scientific Publishing Company, Singapore (2005)
21. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood component analysis. In: NIPS. MIT Press, Cambridge (2005)
22. Woźnica, A., Kalousis, A., Hilario, M.: Matching based kernels for labeled graphs. In: Mining and Learning with Graphs (MLG 2006), with ECML/PKDD, Berlin, Germany (2006)
23. Ramon, J., Bruynooghe, M.: A polynomial time computable metric between point sets. Acta Informatica 37(10), 765–780 (2001)
24. Gärtner, T., Lloyd, J.W., Flach, P.A.: Kernels and distances for structured data. Mach. Learn. 57(3), 205–232 (2004)

A New Framework for Dissimilarity and Similarity Learning

Adam Woźnica and Alexandros Kalousis

University of Geneva, Computer Science Department
7 Route de Drize, Battelle batiment A
1227 Carouge, Switzerland
{Adam.Woznica,Alexandros.Kalousis}@unige.ch

Abstract. In this work we propose a novel framework for learning a (dis)similarity function. We cast the learning problem as a binary classification task or a regression task in which the new learning instances are the pairwise absolute differences of the original instances. Under the classification approach the class label we assign to a specific pairwise difference indicates whether the two original instances associated with the difference are members of the same class or not. Under the regression approach we assign positive target values to the pairwise differences of instances from different classes and negative target values to the differences of instances of the same class. The computation of the (dis)similarity of two examples amounts to the computation of prediction scores for classification, or the prediction of a continuous value for regression. The proposed framework is very general as we are free to use any learning algorithm. Moreover, our formulation generally leads to a (dis-)similarity which, depending on the learning algorithm, can be efficient and simple to learn. Experiments performed on a number of classification problems demonstrate the effectiveness of the proposed approach.

1 Introduction

The k-Nearest Neighbour (kNN) algorithm is an effective method to address classification problems that has proved its utility in many real-world applications [1]. Most common kNN classifiers use the Euclidean metric to measure the dissimilarities between examples. This approach has the advantages of simplicity and generality, however, its main limitation is that the Euclidean metric implies that the input space is isotropic which is rarely valid in practical applications [2].

Since the Euclidean metric is not appropriate for many real-world learning problems different researchers have recently proposed methods for learning the parameters of a parametrized distance measure directly from the data, either in a fully supervised setting [2–7] or using side information [8–10]. The distances in the above methods are usually restricted to belong to a Mahalanobis metric family parametrized by a positive semi-definite (PSD) matrix $\boldsymbol{A}$.[1] The goal in metric learning is to discover an "optimal" matrix $\boldsymbol{A}$ that achieves a higher kNN predictive performance than the Euclidean metric.

[1] The Mahalanobis metric is defined as: $d_{\boldsymbol{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{A}(\boldsymbol{x}_i - \boldsymbol{x}_j)$.

In most of the above metric learning methods the input information is given in the form of equivalence relations. A widely used equivalence relation in the classification setting indicates whether two instances, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, belong to the same class or not. It is important to realize that the existing techniques do not require a direct access to the instances, instead they access them through their pairwise distances, or equivalently through their pairwise difference vectors $|\boldsymbol{x}|_{ij}$. The elements of the latter are the absolute differences of the attributes of the two instances. More formally, for $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^p$, $|\boldsymbol{x}|_{ij}$ is defined as:

$$|\boldsymbol{x}|_{ij}^T = (|x_{i_1} - x_{j_1}|, \ldots, |x_{i_p} - x_{j_p}|)^T \tag{1}$$

where x_{i_l} denotes the l-th attribute of $\boldsymbol{x}_i$. Metric learning algorithm assign weights to the attributes vectors of the form of (1), or to their pairwise products in the case of quadratic metrics. The weighted differences are then aggregated to compute the final Mahalanobis metric. In general, the metric learning methods assign the weights so that under the new metric pairs of instances of the same class will be close together (i.e. their Mahalanobis metric will have a value close to 0), and pairs of instances of different classes will be far apart (i.e. their Mahalanobis metric will have larger values).

Based on the above observations we go one step further and use the pairwise difference vectors of the form given by equation 1 as our learning instances. More precisely, our approach is based on casting the dissimilarity learning problem as a binary classification or a regression task defined over the space of the absolute difference vectors. When we treat the problem as a classification task we assign negative labels to these difference vectors that correspond to pairs of instances of the same class and positive labels to the difference vectors that correspond to pairs of instances from different classes. In the regression scenario we assign negative and positive target values, respectively. The construction of the learning problem in this new space is justified by the fact that for instances of the original space that belong to the same class, some attributes of $|\boldsymbol{x}|_{ij}$ should have small values, while for instances of different classes these attributes should have large values. Moreover, by exploring classification or regression algorithms that produce models based on weighted combinations of the input attributes (e.g. Support Vector Machines or Logistic Regression), we expect that the non-discriminatory attributes will be assigned low weights, and hence instances in the new space with positive and negative classes (or positive and negative numbers) will be moved respectively far from and towards the origin of the new space. In our framework the computation of a dissimilarity measure between two examples amounts to computing a prediction score in classification or a continuous value in regression; the higher the predicted score or the predicted value of the target variable, the more dissimilar are the corresponding two input instances.

The paper is organized as follows. In Sect. 2 we present the existing metric learning techniques under a common framework. This will directly motivate the main contribution of this work presented in Sect. 3, where we propose a new framework for learning (dis-)similarity measures. Experimental results are reported in Sect. 4. Finally, we conclude with Sect. 5 where we discuss major open issues and future work.

2 Metric Learning

In this section we will present the metric learning problem in a common framework that will help us to motivate the main contribution of this work in Sect. 3. We begin with a labeled set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\} = (\mathcal{X}, \mathcal{Y})$ where $\boldsymbol{x}_i \in \mathbb{R}^p$ and $y_i \in \{1, 2, \ldots, c\}$.

Based on the notation from (1), we will represent in a compact way both Euclidean and Mahalanobis metrics between two instances $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^p$. More precisely, the squared Euclidean metric can be defined as $d^2(\boldsymbol{x}_i, \boldsymbol{x}_j) = |\boldsymbol{x}|_{ij}^T |\boldsymbol{x}|_{ij}$ while the squared Mahalanobis metric, parameterized by a positive semi-definite matrix $\boldsymbol{A} \in \mathbb{R}^{p \times p}$ ($\boldsymbol{A} \succeq 0$), can be represented as:

$$d_{\boldsymbol{A}}^2(\boldsymbol{x}_i, \boldsymbol{x}_j) = |\boldsymbol{x}|_{ij}^T \boldsymbol{A}\, |\boldsymbol{x}|_{ij}. \tag{2}$$

We note that it is sometimes useful to reparametrize the Mahalanobis metric as:

$$d_{\boldsymbol{W}}^2(\boldsymbol{x}_i, \boldsymbol{x}_j) = |\boldsymbol{x}|_{ij}^T \boldsymbol{W}^T \boldsymbol{W}\, |\boldsymbol{x}|_{ij} \tag{3}$$

where $\boldsymbol{A} = \boldsymbol{W}^T\boldsymbol{W}$ and $\boldsymbol{W} \in \mathbb{R}^{p \times p}$. For any $\boldsymbol{W}$ we have $\boldsymbol{A} = \boldsymbol{W}^T\boldsymbol{W} \succeq 0$. In what follows, to emphasize that all the above metrics between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ depend only on $|\boldsymbol{x}|_{ij}$, we will denote $d^2(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $d_{\boldsymbol{L}}^2(\boldsymbol{x}_i, \boldsymbol{x}_j)$ ($\boldsymbol{L} = \boldsymbol{A}$ or $\boldsymbol{L} = \boldsymbol{W}$) as $d^2(|\boldsymbol{x}|_{ij})$ and $d_{\boldsymbol{L}}^2(|\boldsymbol{x}|_{ij})$, respectively.

A common approach in the existing metric learning methods is to provide information in the form of equivalence relations as pairwise constraints on the input instances. In the classification framework there is a natural equivalence relation, namely whether two vectors share the same class label or not, i.e. $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) : c_{ij} = 0\}$ and $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) : c_{ij} = 1\}$ where $c_{ij} \in \{0, 1\}$ indicates whether or not the labels y_i and y_j match:

$$c_{ij} = \begin{cases} 0 & \text{if } y_i = y_j \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

It should be stressed that the existing distance learning methods do not require a direct access to the pairs of instances in either of the above sets $\mathcal{S}$ and $\mathcal{D}$; instead, they access the data through the distance functions $d_{\boldsymbol{A}}$ or $d_{\boldsymbol{W}}$ and hence only though pairwise distance vectors $|\boldsymbol{x}|_{ij}$ of the form given in (1). Consequently, to emphasize the above fact, we will consider only the following versions of the equivalence relations $\mathcal{S}' = \{|\boldsymbol{x}|_{ij} : c_{ij} = 0\}$ and $\mathcal{D}' = \{|\boldsymbol{x}|_{ij} : c_{ij} = 1\}$.

The general problem of metric learning in a supervised setting can be now stated as the following optimization problem:

$$\min_{\boldsymbol{L}}.\ \mathcal{F}_{\boldsymbol{L}}(\mathcal{S}', \mathcal{D}') + \lambda \Omega(\boldsymbol{L}) \tag{5}$$

where $\mathcal{F}_{\boldsymbol{L}}$ is a (possibly non-differentiable) cost function, $\boldsymbol{L} = \boldsymbol{A}$ or $\boldsymbol{L} = \boldsymbol{W}$ and $\Omega(\cdot)$ is a regularization term[2] whose importance is controlled by the λ regularization parameter. Additionally, the above optimization problem is possibly subject to a number

[2] We set $\Omega(\boldsymbol{A}) = Tr(\boldsymbol{A})$ and $\Omega(\boldsymbol{W}) = \|\boldsymbol{W}\|_F^2$, where $Tr(\cdot)$ and $\|\cdot\|_F$ denote the matrix trace and the Frobenious norm, respectively.

of constraints. For example, for the parametrization from (2) the optimization given in (5) has to be constrained by $\boldsymbol{A} \succeq 0$. Depending on the actual form of the function $\mathcal{F}_{\boldsymbol{L}}$ different instantiations of the algorithm can be obtained.

One possible problem with the optimization problem of (5) is that for full matrices $\boldsymbol{L}$ the number of parameters to estimate is p^2. For large p (i.e. in the order of few thousands), this would render the optimization task non tractable as there will be too many parameters to optimize over. We explore a solution to this problem that is based on restricting matrices $\boldsymbol{L}$ to be diagonal, resulting in a weighted combination of features (this restriction can be seen as a simple form of regularization since it reduces the effective number of parameters from p^2 to p). It should be noted that the approach based on diagonal matrices, although faster then the one based on full matrices, is also less expressive since it does not account for interactions between different attributes. On the other hand, it allows for the application of metric techniques also on high-dimensional datasets.[3]

In the rest of this section we will present 3 different instantiations of the above framework which differ with respect to the objective function $\mathcal{F}_{\boldsymbol{L}}$ and hence the assumptions they make for the data distribution. More precisely, in this work we will focus on the following state-of-the-art metric learning algorithms: Large Margin Nearest Neighbor (LMNN) [11], Maximally Collapsing Metric Learning (MCML) [4] and Neighborhood Component Analysis (NCA) [3].

LMNN. The cost function $\mathcal{F}_{\boldsymbol{A}}$ of LMNN [2] is constructed in such a way that it penalizes both large distances between each sample and its similarly labeled nearest neighbors, and small distances between differently labeled instances. Equivalently, the criterion of LMNN seeks for a metric in which each sample has a large margin between nearest neighbors of same class and samples in different classes. We use $\eta_{ij} \in \{0, 1\}$ to denote the neighbourhood relation between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ where $\eta_{ij} = 1$ indicates the sample $\boldsymbol{x}_i$ is one of the neighbors of sample $\boldsymbol{x}_j$, and $\eta_{ij} = 0$ otherwise. The LMNN method can be now formulated as the following optimization problem $\min._{\boldsymbol{A}} \ \mathcal{F}_{\boldsymbol{A}} = \sum_{ij} \eta_{ij} d^2_{\boldsymbol{A}}(|\boldsymbol{x}|_{ij}) + \mu \sum_{il} \eta_{il}(1 - c_{il})\xi_{ijl}(\boldsymbol{A})$ where $\xi_{ijl}(\boldsymbol{A}) = \max\{0, 1 + d^2_{\boldsymbol{A}}(|\boldsymbol{x}|_{ij}) - d^2_{\boldsymbol{A}}(|\boldsymbol{x}|_{il})\}$, μ is a constant (we fix $\mu = 1$) and $\boldsymbol{A} \succeq 0$.

MCML. The MCML algorithm is based on the simple geometric intuition that all points of the same class should be mapped onto a single location and far from points of the other classes [4]. To learn the metric which would approximate this ideal geometrical setup a conditional distribution is introduced which for each example $\boldsymbol{x}_i$ selects another example $\boldsymbol{x}_j$ as its neighbor with some probability $p_{\boldsymbol{A}}(j|i)$, and $\boldsymbol{x}_i$ inherits its class label from $\boldsymbol{x}_j$. The probability $p_{\boldsymbol{A}}(j|i)$ is based on the softmin of the $d^2_{\boldsymbol{A}}$ distance measure, i.e. $p_{\boldsymbol{A}}(j|i) = \frac{\exp(-d^2_{\boldsymbol{A}}(|\boldsymbol{x}|_{ij}))}{\sum_{k \neq i} \exp(-d^2_{\boldsymbol{A}}(|\boldsymbol{x}|_{ik}))}$, $p_{\boldsymbol{A}}(i|i) = 0$. It can be shown [4] that any set of points which has the distribution $p_0(j|i) \propto 1$ if $|\boldsymbol{x}|_{ij} \in \mathcal{S}'$ and $p_0(j|i) = 0$ if $|\boldsymbol{x}|_{ij} \in \mathcal{D}'$ exhibits the desired ideal geometry. It is thus natural to seek a matrix $\boldsymbol{A}$ such that $p_{\boldsymbol{A}}(\cdot|i)$ is as close (in the sense of the Kullback-Leibler divergence) to $p_0(\cdot|i)$. This, after a number of transformations [4], is equivalent to minimizing $\mathcal{F}_{\boldsymbol{A}} = -\sum_{ij}(1 - c_{ij}) \ln(\frac{\exp(-d^2_{\boldsymbol{A}}(|\boldsymbol{x}|_{ij}))}{\sum_{k \neq i} \exp(-d^2_{\boldsymbol{A}}(|\boldsymbol{x}|_{ik}))})$ subject to $\boldsymbol{A} \succeq 0$.

[3] An alternative approach is to reduce the dimensionality of the input data to p' ($p' \ll p$); we will also consider this solution in Sect. 4.

NCA. The NCA method attempts to directly optimize a continuous version of the leave-one-out error of the kNN algorithm on the training data. The main difference between NCA and the two previous methods is that optimization in NCA is done with respect to matrix $\boldsymbol{W}$ of (3). Its cost function $\mathcal{F}_{\boldsymbol{W}}$ is based on stochastic neighbor assignments in the weighted feature space, which is based on $p_{\boldsymbol{A}}(j|i)$ defined above where $\boldsymbol{A}$ is replaced with $\boldsymbol{W}$. Under this stochastic selection rule the probability $p_{\boldsymbol{W}}(i)$ of correctly classifying $\boldsymbol{x}_i$ is given by $\sum_j (1 - c_{ij}) \frac{\exp(-d^2_{\boldsymbol{W}}(|\boldsymbol{x}|_{ij}))}{\sum_{k \neq i} \exp(-d^2_{\boldsymbol{W}}(|\boldsymbol{x}|_{ik}))}$. In this work the actual function to minimize is $\mathcal{F}_{\boldsymbol{W}} = -\sum_i \ln\{\sum_j (1 - c_{ij}) \frac{\exp(-d^2_{\boldsymbol{W}}(|\boldsymbol{x}|_{ij}))}{\sum_{k \neq i} \exp(-d^2_{\boldsymbol{W}}(|\boldsymbol{x}|_{ik}))}\}$ which expresses the probability of obtaining an error free classification on the training set [3].

3 Scoring Based (Dis-)Similarity Learning

In this section we present the main contribution of this work and define a new framework for (dis-)similarity learning over vectorial data. As already mentioned in Sect. 2, most of the existing metric learning techniques do not require a direct access to the training data; instead, the only "interface" to the data is through the equivalence sets $\mathcal{S}'$ and $\mathcal{D}'$, the elements of which are the pairwise difference vectors as these were defined in equation (1). Motivated by this observation, we will cast the problem of dissimilarity learning as a problem of learning a binary classification scoring (or regression) function from the training data constructed from $\mathcal{S}'$ and $\mathcal{D}'$. In classification the new labels will be negative for elements of $\mathcal{S}'$ and positive for elements of $\mathcal{D}'$. In regression pairs of instances of different and pairs of instances of the same class will be assigned positive and negative numbers, respectively. More formally, the new training data is given as:

$$\mathcal{I} = \{\bigcup_{i>j}^{n} (|\boldsymbol{x}|_{ij}, c_{ij})\} \cup (\mathbf{0}^T, 0) \tag{6}$$

where c_{ij} is defined in (4) and $\boldsymbol{O}$ denotes a vector of zeros; $(\mathbf{0}^T, 0)$ is included in the new training data as it models for the fact that duplicate instances share the same class. It should be noted that the size of the training dataset from (6) scales as $O(n^2)$ (it contains exactly $\frac{(n-1)(n-2)}{2} + 1$ examples). In these settings, the learning phase amounts to training a learning algorithm over the training data $\mathcal{I}$, whereas the computation of the (dis-)similarity between two examples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ boils down to a computing prediction score (for classification) or predicted depended variable (for regression) for an instance $|\boldsymbol{x}|_{ij}$. In the remaining part of this work we will sometimes denote the scoring (or regression) function for $|\boldsymbol{x}|_{ij}$ as $score(\boldsymbol{x}_i, \boldsymbol{x}_j) = score(|\boldsymbol{x}|_{ij})$. The procedure of classifying an instance $\boldsymbol{x}_{new} \in \mathcal{X}$ by the kNN algorithm, that is based on computing $score(|\boldsymbol{x}|_{ij})$ to select nearest neighbours, is presented in Algorithm 1.

It is important to realize that for the definition of c_{ij} from (4), we can interpret the scoring function $score(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as a *dissimilarity measure* since it assigns *lower* values to elements of $\mathcal{S}'$ and *higher* values for elements of $\mathcal{D}'$. However, by redefining c_{ij} from (4) so that it assigns negative labels for elements of $\mathcal{D}'$ and positive labels for elements of $\mathcal{S}'$, the corresponding scoring functions can be interpreted as a *similarity measure*. In this study we will only focus on learning dissimilarity measures.

Algorithm 1. kNN classification using scoring function.

```
1: $\boldsymbol{kNN\_classify}(\boldsymbol{x}_{new}, \mathcal{X}, \mathcal{Y}, \boldsymbol{score}(\cdot), k))$
2: // $\boldsymbol{x}_{new}$: an instance to classify
3: // $\mathcal{X}, \mathcal{Y}$: input data
4: // $\boldsymbol{score}(\cdot)$: a scoring function learned on (6)
5: // $k$: number of nearest neighbours
6: $S \leftarrow \emptyset$
7: for $i \in 1, \ldots, n$ do
8:     $|\boldsymbol{x}|^T_{new,i} \leftarrow (|x_{new_1} - x_{i_1}|, \ldots, |x_{new_p} - x_{i_p}|)^T$
9:     $S \leftarrow S \cup \boldsymbol{score}(|\boldsymbol{x}|_{new,i})$
10: end for
11: $S \leftarrow \boldsymbol{sort}(S, ascend)$
12: $N(\boldsymbol{x}_{new}) \leftarrow k$ nearest neighbors of $\boldsymbol{x}_{new}$ according to $S$
13: $N_c(\boldsymbol{x}_{new}) \leftarrow$ elements of $N(\boldsymbol{x}_{new})$ of class $c$
14: return $\text{argmax}_{c \in C} \sum_{\boldsymbol{x} \in N_c(\boldsymbol{x}_{new})} \delta(class(\boldsymbol{x}), c)$
```

The proposed framework has several advantages over existing metric learning methods. First, it is very general as we are free to use almost any classification (or regression) algorithm as long as its decision is based on a classification score (the predictions of a regression algorithm could be interpreted right away as dissimilarities). The only restriction we put on the learner is that it should scale well with respect to the number of input instances; this is due to the fact that the number of instances in the new training set $\mathcal{I}$ scales as $O(n^2)$, and hence any algorithm applied in the new space, whose computational complexity is higher than say log-linear would be prohibitive but for toy learning problems. Second, unlike most of the existing techniques, in general no semi-definite programming or eigenvalue computations are required, and hence depending on the employed learner the resulting dissimilarity can be efficiently learned. Finally, our formulation generally leads to a dissimilarity that can be more expressive, and at the same time simpler to learn, than the standard Mahalanobis metrics.

We also mention that there are two main drawbacks of our approach. First, in general the learned dissimilarity measures are not valid metrics.[4] However, several authors have reported state-of-the-art classification performance of kNN over a variety of learning problems where the underlying distance measures were not valid metrics, see e.g. [12, 13]. In particular, most of the examined non-metric distance measures do not satisfy the triangle inequality; the latter guarantees that if a point $\boldsymbol{x}_i$ is close to $\boldsymbol{x}_j$ and close to $\boldsymbol{x}_l$ then $\boldsymbol{x}_j$ will be also close to $\boldsymbol{x}_k$. Moreover, as we will see in the experimental part of this study, the kNN algorithm, where the nearest neighbors are selected using $score(|\boldsymbol{x}|_{ij})$, generally outperforms kNN with adaptive metrics that are learned using different state-of-the-art metric learning techniques. This might suggest that the metric conditions of a dissimilarity function are not necessary for kNN to achieve good predictive performance. However, the dissimilarity measures that are not metrics might be

[4] Technically, a function $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a metric if it is: (i) non-negative ($d(x, y) \geq 0$), (ii) reflexive ($d(x, x) = 0$), (iii) strict ($d(x, y) = 0 \Rightarrow x = y$), (iv) symmetric ($d(x, y) = d(y, x)$) and (v) satisfies triangle inequality ($d(x, z) \leq d(x, y) + d(y, z)$), $\forall_{x,y,z \in \mathcal{X}}$.

not adequate for some applications.[5] We will discuss both the forms and characteristics of our dissimilarities in the remainder of this section.

The second, and potentially more severe, problem is that the learning instances from (6) are not independent. This renders the application of the learning algorithms in the new space questionable, as the basic assumption that training instances should be independently and identically distributed (i.i.d) does not hold. However, the good experimental performance of our framework reported in Sect. 4 suggests that this difficulty is lifted by the above mentioned flexibility of learned dissimilarities. In other words, the advantage of using very flexible dissimilarities might overcome the problem of non-independently distributed data.

In the remainder of this section we will describe 3 different instantiations of our framework that differ with respect to the employed learning algorithm. As already mentioned, the two requirements we put on the learning algorithms applied in the new space are that they should (i) output a function indicating how similar are two instances and (ii) scale well with respect to the number of instances in $\mathcal{I}$. To perform a fair comparison with the existing metric learning techniques we will only focus on algorithms that produce linear models whose parameters are directly related with the parameters of the Mahalanobis metric. Based on the above considerations, in this study we focused on two classification (Linear Support Vector Machines and Logistic Regression) and one regression algorithm (Ridge Regression) which fulfil the above requirements.

Support Vectors Machines. In Linear Support Vector Machines (L-SVM) the learning phase amounts to solving the following unconstrained quadratic optimization problem [15]:[6] $\min_{\boldsymbol{w}\in\mathbb{R}^p} \sum_{ij} \max\{0, 1 - c_{ij}\langle \boldsymbol{w}, |\boldsymbol{x}|_{ij}\rangle\} + \lambda\|\boldsymbol{w}\|^2$, where λ is a user-defined regularization parameter and $i = 1,\ldots,n; j = i,\ldots,n$. The dissimilarity between $\boldsymbol{x}_i, \boldsymbol{x}_j$ is given as: $score(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{w}^*, |\boldsymbol{x}|_{ij}\rangle$, where $\boldsymbol{w}^*$ is a solution of the optimization problem of SVM. It is easy to verify that the above function is not a metric as it can take negative values and does not satisfy the triangle inequality, however, it is reflexive, strict (assuming a non-degenerate $\boldsymbol{w}^*$) and symmetric. For solving the optimization problem of L-SVM we exploit the algorithm recently proposed in [15] that scales linearly with respect to the number of instances in $\mathcal{I}$, i.e. its computational complexity is $O(n^2)$. It is worth noting that the method from [5] also exploits the notion of SVM to learn a metric. The main difference from our framework is that this method is a *local* one that aims to determine a stretched neighbourhood around each query instance such that class conditional probabilities are likely to be constant. Moreover, [5] exploits the softmax function to obtain a valid metric. We plan to perform a detailed comparison of the two approaches in the future.

Logistic Regression. Logistic Regression (LR) [1] is a well known binary classification method where the classification decisions are based on a scoring function $score(|\boldsymbol{x}|_{ij})$ that can be interpreted as a probability p_{ij} that an instance $|\boldsymbol{x}|_{ij}$ belongs to the positive class. More formally, $score(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is modelled as: $score(\boldsymbol{x}_i, \boldsymbol{x}_j) \equiv p_{ij} = \frac{\exp(\langle \boldsymbol{w}, |\boldsymbol{x}|_{ij}\rangle)}{1+\exp(\langle \boldsymbol{w}, |\boldsymbol{x}|_{ij}\rangle)}$,

[5] We also note that the metric conditions are crucial if one employs efficient nearest neighbour search strategies that are based on storing training examples in hierarchical data structures [14].

[6] As we will exploit the learned weights only to compute a scoring function we do not include in L-SVM (and in Logistic Regression) the bias term b.

where $\boldsymbol{w} \in \mathbb{R}^p$ are parameters of the logistic regression model; $score(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is non-negative and symmetric, but does not satisfy the triangle inequality and is neither strict nor reflexive. We will exploit the regularized version of LR where the optimal solution $\boldsymbol{w}^*$ is obtained by solving the following optimization problem: $\min._{\boldsymbol{w}\in\mathbb{R}^p} \; -\sum_{ij}(1 - c_{ij}) \ln(\frac{\exp(\langle \boldsymbol{w}, |\boldsymbol{x}|_{ij}\rangle)}{1+\exp(\langle \boldsymbol{w}, |\boldsymbol{x}|_{ij}\rangle)}) + \lambda\|\boldsymbol{w}\|^2$.

Ridge Regression. We also experimented with one regression method, namely the Ridge Regression (RR) algorithm [1] that solves the following optimization problem: $\min._{\boldsymbol{w}\in\mathbb{R}^p} \; \sum_{ij}(\langle \boldsymbol{w}, |\boldsymbol{x}|_{ij}\rangle - c_{ij})^2 + \lambda\|\boldsymbol{w}\|^2$. In this context, the dissimilarity function has an identical form (and properties) as in the case of L-SVM $score(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{w}^*, |\boldsymbol{x}|_{ij}\rangle$, where $\boldsymbol{w}^* \in \mathbb{R}^p$ is a solution of the optimization problem.

4 Experiments

We evaluated the performance of the proposed approach on a number of real-world classification problems. The goal is to examine whether the three instantiations of our dissimilarity learning framework from Sect. 3 (i.e. the L-SVM, LR and RL linear algorithms) achieve better predictive performance than a number of existing metric learning algorithms. The quality of the different dissimilarity measures will be only compared in the context of kNN (we will not use the underlying algorithms such as logistic regression directly for classification).

We compared the above 3 methods with the LMNN, MCML and NCA state-of-the-art metric learning algorithms. We experimented with 2 instantiations of the above metric learning techniques, over full and diagonal matrices denoted respectively as $\mathrm{METHOD}_{\mathrm{full}}$ and $\mathrm{METHOD}_{\mathrm{diag}}$, where METHOD is LMNN, MCML or NCA. For comparison reasons we also provide the performance of the standard kNN algorithm with the Euclidean metric. We experimented with different values of k ($k = 1, 3, 10$); the relative performance of the different methods did not vary with k, we report results only for $k = 1$. In all the above methods we set the λ parameter to 1. In all the experiments we estimate accuracy using 10-fold cross-validation and control for the statistical significance of observed differences using McNemar's test [16] (sig. level of 0.05).

We experimented with 13 datasets. First, we used 4 standard datasets from the UCI repository (Liver, Wdbc, Wine, BalanceScale); these datasets are standard benchmarks used in the context of distance learning. Then, we have chosen to experiment with high-dimensional data from two different application domains, namely genomics and proteomics (description of these datasets can be found in [17]). The genomics datasets correspond to DNA-microarray experiments. We worked with three different datasets: colon cancer (Colon), central nervous system (Central) and Leukemia (Leuk). All proteomics datasets come from the domain of mass spectrometry. We worked with 4 different datasets: ovarian cancer (Ovarian), prostate cancer (Prostate), an early stroke diagnosis (Stroke), and MaleFemale (MaleF). In Ovarian, Prostate and Stroke we experimented with 2 versions of each of the above proteomics datasets where we used different parameters in the prepossessing step for feature extraction (in MaleF we had access only to one version of this dataset). All features correspond to intensities of mass values and are continuous. All the above genomics and proteomics datasets, in addition

Table 1. Accuracy and significance test results of kNN (with and without distance learning). The numbers in parentheses indicate the number of significance points that the algorithm scores in a given dataset; the algorithms with the highest score for each dataset are highlighted. n and p denote respectively the number of instances and the dimensionality of the data.

Dataset	n	p	LMNN$_{\text{diag}}$	MCML$_{\text{diag}}$	NCA$_{\text{diag}}$	L-SVM	LR	RR	kNN
Liver	345	6	58.6 (3.5)	**62.4 (4.0)**	54.4 (2.0)	57.5 (3.5)	58.1 (3.5)	47.7 (0.5)	**62.4 (4.0)**
Wdbc	569	30	**94.8 (3.5)**	**94.8 (3.5)**	**95.0 (3.5)**	**93.4 (3.5)**	84.5 (1.0)	71.6 (2.5)	95.5 (3.5)
Wine	178	13	**96.7 (4.5)**	94.9 (4.0)	91.7 (3.0)	**97.4 (4.5)**	73.3 (1.0)	55.2 (0.0)	94.9 (4.0)
Balance	625	4	77.7 (3.0)	77.4 (3.0)	75.5 (2.5)	76.0 (2.5)	76.2 (2.5)	76.1 (3.0)	**79.4 (4.5)**
Colon	62	2000	82.1 (3.0)	80.4 (3.0)	77.5 (2.5)	85.4 (3.5)	85.4 (3.5)	**87.1 (4.0)**	73.8 (1.5)
Central	60	7129	60.0 (3.0)	56.7 (3.0)	48.3 (2.0)	**66.7 (3.5)**	**70.0 (3.5)**	61.7 (3.0)	56.7 (3.0)
Leuk	72	7129	**97.1 (4.0)**	**97.1 (4.0)**	87.5 (1.0)	95.7 (3.5)	**97.1 (4.0)**	**97.1 (4.0)**	84.9 (0.5)
MaleF	134	1524	77.6 (4.0)	68.8 (2.5)	55.7 (0.5)	**84.2 (4.5)**	**83.4 (4.5)**	**81.9 (4.5)**	55.5 (0.5)
Ovarian1	253	385	**96.8 (4.5)**	93.0 (1.0)	91.5 (1.0)	**97.6 (4.5)**	**98.0 (4.5)**	**96.8 (4.5)**	92.2 (1.0)
Ovarian2	253	10361	95.7 (4.5)	86.6 (1.0)	88.0 (1.0)	95.7 (4.5)	95.0 (3.5)	**97.6 (5.0)**	90.6 (1.5)
Stroke1	208	172	**68.6 (4.0)**	63.9 (3.0)	58.1 (2.5)	60.4 (2.5)	63.9 (3.0)	60.7 (3.0)	64.1 (3.0)
Stroke2	208	2810	**69.0 (4.0)**	60.1 (1.5)	63.5 (3.0)	**70.7 (4.0)**	68.7 (3.5)	**70.1 (4.0)**	55.5 (1.0)
Prostate1	322	390	85.8 (5.0)	78.3 (2.0)	70.8 (0.5)	**86.1 (5.5)**	82.6 (3.5)	78.6 (2.0)	82.4 (2.5)
Prostate2	322	12600	90.3 (3.5)	83.3 (1.0)	85.0 (2.5)	92.3 (4.0)	92.0 (4.0)	**94.2 (4.5)**	82.2 (1.5)
Average rank			3.86	2.61	1.96	3.86	3.29	3.14	2.29

to large number of features, are also characterized by a small number of observations, making these datasets a difficult learning scenario. In all the above datasets the numeric attributes were normalized so that they takes values between 0 and 1. In Table 1 we provide the number of instances and attributes in the examined datasets.

To better understand the relative performances of the examined algorithms we established a ranking schema of these algorithms based on the results of the pairwise comparisons. More precisely, if an algorithm is significantly better than another it is credited with 1 point; if there is no significant difference between two algorithms then they are credited with 0.5 points; if an algorithm is significantly worse than another it is credited with 0 point. Thus, in the case m algorithms are examined, an algorithm that is significantly better than all the others for a given dataset is assigned a score of $m - 1$.

Experiments on these datasets have 2 goals. First, we study the relative performance of our methods with the existing metric learning algorithms. In these experiments we use the diagonal versions of the existing metric learning techniques as it allows for a fair comparison with the proposed framework; similar to the metric learning techniques based on diagonal matrices, L-SVM, LR and RR do not account for interactions between different attributes. Second, we compare the predictive performance of our method with the metric learning methods based on full matrices. For the latter methods applied on the high-dimensional datasets (i.e. all the genomics and proteomics classification problems) we exploited the PCA method to reduce the data dimensionality; this procedure was widely used in the context of metric learning [2]. More precisely, the training instances are projected into a lower dimensional subspace accounting for at least 99 $\%$ of the data's total variance.

Results and Analysis. In Table 1 we present the results (with the score ranks) of the comparison between $\mathrm{LMNN}_{\mathrm{diag}}$, $\mathrm{MCML}_{\mathrm{diag}}$, $\mathrm{NCA}_{\mathrm{diag}}$, L-SVM, LR, RR and the standard kNN (recall that the maximum score for an algorithm in a given dataset is 6). From these results we can make several observations. First, with the exception of the Liver and Balance datasets, there is at least one adaptive method that outperforms the standard kNN algorithm (in Liver and Balance, kNN is placed in the first position according to our ranks). Second, the developed dissimilarity methods based on L-SVM, LR and RR generally outperform both $\mathrm{MCML}_{\mathrm{diag}}$ and $\mathrm{NCA}_{\mathrm{diag}}$; the advantage of our methods is most visible in the genomics and proteomics high dimensional datasets. Finally, the methods that most often win are $\mathrm{LMNN}_{\mathrm{diag}}$, L-SVM, LR and RR.

To quantify the relative performances of the different algorithms we computed for each method its average rank over all the examined datasets. These ranks are presented in the last row of Table 1. We observe that the best performance of 3.86 points is obtained for the margin based methods (L-SVM and $\mathrm{LMNN}_{\mathrm{diag}}$), which have a similar computational complexity both in theory (they scale as $O(n^2)$) and in practice (they had similar running times). This result is remarkable since L-SVM, which is based on a simple idea, performs equally well as the more elaborate metric learning algorithm that has been reported to consistently outperform other metric learning techniques over a number of of non-trivial learning problems [2]. Finally, we mention that the surprisingly poor performance of $\mathrm{NCA}_{\mathrm{diag}}$ might be explained by the fact that its cost function is not convex and hence it is sensitive to the initializations of $\boldsymbol{W}$.

In the second set of experiments we compare the performance of the metric learning methods with the full matrix to that of metric learning with diagonal matrix. Here we want to examine whether methods that account for feature interaction outperform the methods proposed in Sect. 3. As already mentioned, we first reduced the dimensionality of all the high-dimensional datasets by mapping the instances to lower dimensional subspaces defined by the PCA method; depending on the datasets, the dimensionalities of the subspaces, that account for at least 99 % of the data's total variance, were between 50 and 178. The results are presented in Table 2. From these results we can see that with the exception of $\mathrm{NCA}_{\mathrm{full}}$ in Liver and Prostate1, all the metric learning with full matrix have similar or worse performances than their corresponding versions with diagonal matrices (the first signs in parenthesis are mostly "=" or "-"). This could suggest that even though the data dimensionality is significantly reduced, the metric learning algorithms based on full matrices might still be prone to overfitting (the other explanation might be simply that by removing features that have low variance we also remove important discriminatory information).

We have also compared the performances of full matrix metric learning methods with that of L-SVM, LR and RR; the significance tests results corresponding to this comparison are given by the 2nd, 3th and 4th signs in parenthesis in Table 2. From these results we can see that the relative performances between metric learning methods that are based on full matrices and the methods from Sect. 3 depend on the actual dataset. Indeed, in the first 3 UCI datasets (Liver, Wdbc and Wine) there is an advantage of full matrix metric learning algorithm; in Balance, Central and Stroke1 there is almost no difference in performances; in Prostate1 no conclusions can be drawn; and in all the remaining datasets the L-SVM, LR and RR methods generally outperform

Table 2. Accuracy and significance test results of the algorithms based on full matrices. The 4 signs in parenthesis correspond to a comparison between $\mathrm{METHOD}_{\mathrm{full}}$ and $\mathrm{METHOD}_{\mathrm{diag}}$, L-SVM, LR and RR, respectively (the "+" sign stands for a significant win of a the first algorithm in a pair, "-" for a significant loss, and "=" for no significant difference).

Dataset	$\mathrm{LMNN}_{\mathrm{full}}$	$\mathrm{MCML}_{\mathrm{full}}$	$\mathrm{NCA}_{\mathrm{full}}$	Dataset	$\mathrm{LMNN}_{\mathrm{full}}$	$\mathrm{MCML}_{\mathrm{full}}$	$\mathrm{NCA}_{\mathrm{full}}$
Liver	55.4 (====)	61.5 (===+)	62.4 (+==+)	MaleF	67.4 (- - - -)	60.3 (= - - -)	57.1 (=- - -)
Wdbc	94.8 (==+=)	94.8 (==+=)	95.5 (==+=)	Ovarian1	93.4 (- - - =)	87.8 (- - - -)	91.8 (=- - -)
Wine	95.5 (==++)	96.8 (==++)	94.9 (==++)	Ovarian2	97.2 (====)	74.7 (- - - -)	90.5 (=- =-)
Balance	77.6 (====)	77.4 (====)	76.1 (====)	Stroke1	66.1 (====)	57.6 (====)	65.1 (====)
Colon	66.7 (- - - -)	65.0 (- - - -)	72.1 (=- - -)	Stroke2	58.7 (- - - -)	55.5 (=- - -)	56.4 (=- - -)
Central	60.0 (====)	58.3 (====)	55.0 (==- =)	Prostate1	83.6 (==+=)	78.3 (=- - =)	83.0 (+=+=)
Leuk	83.2 (- - - -)	83.2 (- - - -)	84.9 (=- - -)	Prostate2	84.2 (=- - -)	72.5 (- - - -)	82.2 (=- - -)

the metric learning techniques based on full matrices. These results suggest that in the majority of the high dimensional datasets, the feature interactions are not important, and hence the methods that do not account for feature interactions have in general better performances. Alternatively, it might suggest that stronger regularization is needed. Moreover, it is interesting to note that the cases for which the full matrix metric learning methods are good are mostly the UCI datasets that correspond to rather not difficult classification problems. This hints that there might be a bias of method development towards methods that perform well on UCI datasets; however, one can argue that they are really representative of the real world.

5 Conclusions and Future Work

In this paper we prosed a novel framework for learning a (dis-)similarity function over vectorial data, where the learning problem is cast as a binary classification or regression task defined over the space of pairwise differences of the input instances. Our approach generally leads to adaptive (dis-)similarities that are not valid metrics; however, we argue that by learning (dis-)similarities that do not fulfil metric conditions (and are not of the Mahalanobis type) we might have more freedom in adapting these (dis-)similarities for a given problem. Our claim is supported by experimental evidence that, in terms of predictive performance, shows that our framework compares favourably with alternative state-of-the-art distance learning techniques which are trained to learn both full and diagonal Mahalanobis metrics.

In the future we want to exploit other learning techniques applied over the new data representation (e.g. decision trees). We also plan to investigate a Non-Linear version of Support Vectors Machines (NL-SVM) to learn the higher-order interaction between features, similar to that modelled in the Mahalanobis metric. This can be achieved by exploiting the polynomial kernel of degree 2 that induces a feature space that is indexed by all the monomials of degree 2. Since we will not be able to rely on efficient implementation of L-SVM, the main challenge here is to make NL-SVM scalable and practical. Moreover, we plan to test more carefully the pros and cons of the L-SVM and

LMNN methods that in our experiments had similar performance and outperformed other algorithms. Finally, we would like to compare our framework with the approach from [5] as the latter also exploits the notion of SVM to locally learn a metric.

Acknowledgments. This work was partially funded by the European Commission through EU projects DebugIT (FP7-217139) and e-LICO (FP7-231519). The support of the Swiss NSF (Grant 200021-122283/1) is also gratefully acknowledged.

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
2. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. J. Mach. Learn. Res. 10, 207–244 (2009)
3. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood component analysis. In: NIPS. MIT Press, Cambridge (2005)
4. Globerson, A., Roweis, S.: Metric learning by collapsing classes. In: Weiss, Y., Schölkopf, B., Platt, J. (eds.) NIPS, vol. 18, pp. 451–458. MIT Press, Cambridge (2006)
5. Domeniconi, C., Gunopulos, D.: Adaptive nearest neighbor classification using support vector machines. In: NIPS, vol. 14. MIT Press, Cambridge (2002)
6. Davis, J., Kulis, B., Jain, P., Sra, S., Dhillon, I.: Information-theoretic metric learning. In: Proc. 24th International Conference on Machine Learning, ICML (2007)
7. Hastie, T., Tibshirani, R.: Discriminant adaptive nearest neighbor classification and regression. In: NIPS, vol. 8 (1996)
8. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning with application to clustering with side-information. In: NIPS, vol. 15, pp. 505–512. MIT Press, Cambridge (2003)
9. Hertz, T., Bar-Hillel, A., Weinshall, D.: Boosting margin based distance functions for clustering. In: ICML'04, p. 50. ACM Press, New York (2004)
10. Schultz, M., Joachims, T.: Learning a distance metric from relative comparisons. In: Advances in Neural Information Processing Systems, vol. 16. MIT Press, Cambridge (2004)
11. Weinberger, K.Q., Saul, L.K.: Fast solvers and efficient implementations for distance metric learning. In: International Conference on Machine Learning, ICML (2008)
12. Woźnica, A., Kalousis, A., Hilario, M.: Distances and (indefinite) kernels for sets of objects. In: The IEEE International Conference on Data Mining (ICDM), Hong Kong (2006)
13. Horvath, T., Wrobel, S., Bohnebeck, U.: Relational instance-based learning with lists and terms. Machine Learning 43(1/2), 53–80 (2001)
14. Liu, T., Moore, A.W., Gray, A.: New algorithms for efficient high-dimensional nonparametric classification. J. Mach. Learn. Res. 7, 1135–1158 (2006)
15. Franc, V., Sonnenburg, S.: Optimized cutting plane algorithm for support vector machines. In: ICML '08: Proceedings of the 25th international conference on Machine learning (2008)
16. McNemar, Q.: Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika 12, 153–157 (1947)
17. Kalousis, A., Prados, J., Hilario, M.: Stability of feature selection algorithms: a study on high-dimensional spaces. Knowledge and Information Systems 12, 95–116 (2006)

Semantic-Distance Based Clustering for XML Keyword Search

Weidong Yang and Hao Zhu

School of Computer Science, Fudan University,
Handan Road 220, Shanghai, China

Abstract. XML Keyword Search is a user-friendly information discovery technique, which is well-suited to schema-free XML documents. We propose a novel scheme for XML keyword search called XKLUSTER, in which a novel semantic-distance model is proposed to specify the set of nodes contained in a result. Based on this model, we use clustering approaches to generate all meaningful results in XML keyword search. A ranking mechanism is also presented to sort the results.

Keywords: XML, Keyword Search, Clustering.

1 Introduction

Keyword search in database is gaining a lot of attentions [1-11]. Many existing methods about XML keyword search are based on LCA (lowest common ancestor) model. A typical one of them is the SLCA (smallest LCA) method [2, 3, 5], which returns a group of smallest answer subtrees of an XML tree. A smallest answer subtree is defined as: a subtree which includes all the keywords and any subtree of it doesn't contain all the keywords. The root of a smallest answer subtree is called a SLCA. However, SLCA method probably omits some meaningful results. Example 1.1 shown in Fig. 1 illustrates a tree structure of an XML document, in which every node is denoted as its label and the number below it is the Dewey number.

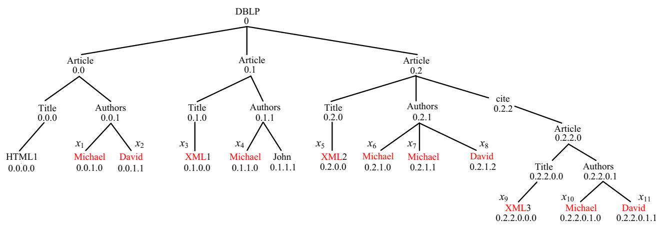


Fig. 1. Tree Structure of an XML document (Example 1.1)

Suppose that keywords are “XML”, “Michael” and “David” are submitted to search in the document, and the nodes containing keywords are marked in red. If SLCA method is applied, the result would be a subtree rooted at node

Article (0.2.2.0). The searching semantics is most likely to be "which papers about XML have been written by Michael and David", and obviously it finds the paper "XML3" while omitting the paper "XML2". Moreover, users may have other reasonable searching intentions, like "which papers are written by Michael and David" and "which papers about XML are written by Michael OR David". For these two requests, paper "HTML1" and "XML1" are both contented results but also omitted. If we regard any node set which contains all the keywords as a candidate results, actually what SLCA method does is picking out some "optimal" ones from all the candidate results, and it considers a candidate result to be "optimal" when its LCA is relatively lowest, so those candidate results with higher LCAs are discarded, this is the essential reason for SLCA method losing some meaningful results. Besides, SLCA method still has some other drawbacks: (1) all the results are isolated, which means any node only can exist in one result, and apparently it is inappropriate in some situations; (2) each result is demanded to contain all the keywords, means that SLCA methods only support *"AND"* logic between all keywords.

This paper presents a novel approach based on clustering techniques for XML keyword search, which can solve all above problems. Main contributions include: We propose a novel semantic distance model for XML keyword search in section 2; three clustering algorithms are designed in section 3, which are graph-based (GC), core-driven (CC) and loosened core-driven (LCC) algorithms; a ranking mechanism is proposed to sort all the searching results in section 4.

2 Search Semantics and Results

We define the keywords submitted by users as a set containing t keywords $L = \{k_i | i = 1, \ldots, t\}$,and the XML document as an XML document tree.

Definition 2.1 (Search Semantics and Results). An XML document tree is a tree coded by Dewey numbers, and denoted as:$d = (V, E, X, label(id), pl(id_1, id_2), depth(id), dwcode(id), lca(V'))$, in which: (1) V is the set of all the nodes, in which each node corresponds to an element or an attribute or a value in the document and has a unique identifier as well as a Dewey number; (2) $E \subseteq V \times V$, is the set of edges in the tree; (3) *label(id)*, is a function which can get the label of the node whose identifier is id;(4)$X \subseteq V$, is the set of keyword nodes in the tree, and a keyword node is a node whose label contains any keyword; (5)$pl(id_1, id_2)$, is a function to obtain the path length between two nodes, in which id_1 and id_2 must have an ancestor-descendant relation, and the function returns the number of edges between them; (6) $depth(id)$ is a function to get the depth of the node whose identifier is id (the depth of the root is 1); (7) $dwcode(id)$, is a function to get the Dewey number of the node whose identifier is id; (8) $lca(V')$, is a function to get the LCA of all nodes in V', and $V' \subseteq V$.

Definition 2.2 (Shortest Path). The shortest path between two nodes The shortest path between two nodes x_i and x_j is the sum of the path between x_i and $lca(x_i, x_j)$ and the path between x_j and $lca(x_i, x_j)$. Meanwhile, the function $spl(x_i, x_j)$ is used to obtain the length of the shortest path. Apparently, $spl(x_i, x_j) = pl(lca(x_i, x_j), x_i) + pl(lca(x_i, x_j), x_j)$.

Definition 2.3 (Semantic Distance). The semantic distance between any two keyword nodes x_i and x_j is defined as a function $dis(x_i, x_j)$: $dis(x_i, x_j) = \frac{spl(x_i,x_j)}{depth(lca(x_i,x_j))}$.

In the rest, semantic distance is called distance for short. In the formula of the distance function, the numerator and the denominator are the length of shortest path and the depth of LCA respectively. Obviously, the semantic distance of two keyword nodes is smaller when their shortest path is shorter or the depth of their LCA is lower. Assume the height (maximum depth) of the tree is h, then the range of $spl(x_i, x_j)$ is $[0, 2h]$ and the range of depth$(lca(x_i, x_j))$ is $[1, h]$, so the range of $dis(x_i, x_j)$ is $[0, 2h]$. In Example 1.1, evaluating all the distances between any two keyword nodes can get a semantic distance matrix (Fig. 2).

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
x_1	0										
x_2	0.67	0									
x_3	6.00	6.00	0								
x_4	6.00	6.00	2.00	0							
x_5	6.00	6.00	6.00	6.00	0						
x_6	6.00	6.00	6.00	6.00	2.00	0					
x_7	6.00	6.00	6.00	6.00	2.00	0.67	0				
x_8	6.00	6.00	6.00	6.00	2.00	0.67	0.67	0			
x_9	8.00	8.00	8.00	8.00	3.00	3.00	3.00	3.00	0		
x_{10}	8.00	8.00	8.00	8.00	3.00	3.00	3.00	3.00	1.00	0	
x_{11}	8.00	8.00	8.00	8.00	3.00	3.00	3.00	3.00	1.00	0.40	0

Fig. 2. Semantic Distance Matrix (Example 1.1)

From Fig. 2, it can be found that the few smallest distances (0.40 and 0.67) are between two authors of a paper, and any largest distances (8.00) is between titles or authors of different papers which have no citation relation. The values of the distances match the practical meanings. Strictly speaking, the searching intentions of users can never be confirmed accurately; so different than existing researches, we suggest that all keyword nodes are useful more or less and should be included in results. Based on the semantic distance model, we divide the set of keyword nodes X into a group of smaller sets, and each of them is called a "cluster".

Definition 2.4 (Cluster). A cluster C_i is a set of keyword nodes, which means $C_i \subseteq X$; and if $C = \{C_i | i = 1, \ldots, m\}$ is the set of all clusters generated, then $\bigcup_{i=1}^{m} C_i = X$.

We set a distance threshold to confine the size of clusters that the distance of any two keyword nodes in a cluster must be less equal to ω. It actually means, two keyword nodes are regarded semantically related if their semantic distance is less than to ω. Besides, there isn't any restriction for which keywords can exist in one cluster, which means both of the *"AND"* and *"OR"* logic between keywords can be supported. Nevertheless, given a distance threshold, there will be massive clusters satisfying the request, and many of them have inclusion relations with each other, so a concept of *"optimal cluster"* is proposed.

Definition 2.5 (Optimal Cluster). Given a distance threshold ω, a set of keyword nodes $C_i \subseteq X$ is called an *optimal cluster* iff: (1) $\forall x_i, x_j \in C_i, s.t.$ $dis(x_i, x_j) \leq \omega$; (2) $\forall x_a, (x_a \in X) and (x_a \notin C_i), \exists x_b \in C_i$ $s.t.$ $dis(x_a, x_b) > \omega$.

Example 2.1: Assume the value of distance threshold is 2.0, then there are only part of the distances can be retained in the table shown in Fig. 2 (marked in blue), four optimal clusters is easily to be obtained: $C_1 = x_1, x_2, C_2 = x_3, x_4, C_3 = x_5, x_6, x_7, x_8 and C_4 = x_9, x_{10}, x_{11}$, which actually correspond to the contents of four papers respectively in Fig. 1.

Finding the optimal clusters is definitely not the last step. Actually a cluster is only a set of keyword nodes, how to generate enough useful information from clusters is another point. A simple approach is: for each cluster C_i, return the whole subtree rooted at $lca(C_i)$ to users. However many these subtrees will have inclusion relations (for example the subtree rooted at $lca(C_3)$ and the subtree rooted at $lca(C_4)$ in Example 2.1), especially when the depth of some $lca(C_i)$ is high, this kind of problems become serious. Moreover, it would cause some results being too large to be returned and for users to obtain useful information. So, for each *(optimal) cluster*, we generate and return a *"minimum building tree"* of it.

Definition 2.6 (Minimum Building Tree). A minimum building tree (MBT) of any cluster C_i is a tree whose root is $lca(C_i)$ and leaves are all the nodes in $descendants(C_i)$. The function $descendants(C_i)$ returns all the nodes in C_i which doesn't have any descendant node in C_i.

Two extra operations are provided to expand the MBTs: (1) the "expansion" of any node, which would add the subtree rooted at the node into the MBT. For example for the MBT of C_2 in Example 2.1, users can expand the node *Authors(0.1.1)* to get another author of the paper *John (0.1.1.1)*; (2) the "elevation" of the root, which will find the parent node of the MBT root and add it into the MBT, for example for the MBT of C_1 in Example 2.1, users can elevate the root *Authors(0.0.1)* and get its parent node *Article(0.0)*. Thus, users can expand each MBT optionally according to their needs until satisfactory information is found.

Definition 2.7 (XML Keyword Search). XML keyword search is a process that: for a set of keywords L submitted by users and a distance threshold ω set in advance, a set of MBTs are obtained by searching an XML document d; moreover, each MBT could be expanded according to the demands of users.

3 Clustering Algorithms

Since our goal is to divide a set of keyword nodes into groups according to the distances between keyword nodes, if each keyword node is regarded as an object, it is very natural to achieve our goal through clustering techniques, so we develop three clustering algorithms for XML keyword search: GC, CC, and LCC.

Lemma 3.1. For an XML document tree *d*, any number of nodes which have the same depth are picked out, and sorted in preorder to form a list *I*. For any node x_i in I, traverse leftwards (rightwards) from it, then the distance between x_i and the node each time met is non-decreasing.

Lemma 3.2. The list I is defined as the same as in Lemma 3.1, also a similar list I' is defined. Assume that the depth of the nodes in I' is higher (lower) than that of the nodes in I, x_i is an arbitrary node in I, and x_i' is the ancestor (descendant) node of x_i in I' (x_i' doesn't have to exist in I' or even in the tree, we can add one in the corresponding position of I' if it's inexistent). When traverse leftwards (rightwards) from x_i' in I', the distance between x_i and the node each time met is non-decreasing.

Proof: For any node x_j in I', equals to $spl(x_i', x_j)$ plus $spl(x_i, x_i')(spl(x_i', x_j)$ minus $spl(x_i, x_i'))$, and since $spl(x_i, x_i')$ is invariable, $spl(x_i, x_i)$ will become greater along with spl(xi', xj) (spl(xi', xj) is non-decreasing). Otherwise, it is obvious that depth(lca(xi, xj)) is non-increasing, according to the formula of semantic distance, is non-decreasing.

After the ω is set and the keywords are submitted, we traverse the XML tree in preorder, find all the keyword nodes and put them into a hierarchical data structure.

Definition 3.1 (Hierarchical Data Structure). Assume the height of the XML document tree is h; the hierarchical data structure H is a data structure which contains h ordered lists of keyword nodes, and the depth of any node in the ith list is i.

When we traverse the XML tree in preorder, each node met would be added into the end of corresponding list in H if its label contains some keyword. Consequently, after the traversal is finished, H will contain all keyword nodes, and because in any list of it all nodes are added in preorder, Lemma 3.1 and Lemma 3.2 hold true in H.

3.1 GC: Graph-Based Clustering Algorithm for XML Keyword Search

In algorithm GC, firstly we traverse H and connect any two nodes between which the distance is less equal to the distance threshold ω with a link. After that a weighted undirected graph whose vertices are all the keyword nodes is obtained. Afterwards, a graph-partition algorithm is used to find all the maximal complete subgraphs (cliques); apparently the vertex set of each clique is an *optimal cluster*. H is accessed from top to bottom, and for each list, nodes are traversed from left to right. Assume x_i is the node currently being accessed, we check the distances between x_i and some neighbors which are right of or below x_i, and because of the accessing order, the nodes left of or above x_i needn't be considered. Assume x_j is a node right of x_i in the same list, according to Lemma 3.1, $dis(x_i, x_j)$ is non-decreasing when the position of x_j moves rightwards, so if $dis(x_i, x_j)$ is greater than ω, all nodes right of x_j in the same list needn't be regarded.

For the nodes in lower lists, firstly it should be confirm that how many lists should be taken into account. For any node x_j below x_i, $dis(x_i,x_j)=\frac{spl(x_i,x_j)}{depth(lca((x_i,x_j)))} \leq \omega$, also because that $depth(x_i) \geq depth(lca((x_i, x_j)))$, then we can easily get that $\frac{spl(x_i,x_j)}{depth(x_i)} \leq \frac{spl(x_i,x_j)}{depth(lca((x_i,x_j)))} \leq \omega$, which indicates that $spl(x_i, x_j) \leq depth(x_i) \times \omega$, so only lists below x_i need to be taken into account. Obviously, even there exists a descendant of x_i in the $(floor(depth(x_i) \times \omega) + 1)$th lower list, the distance overflows.

For each of these $(floor(depth(x_i) \times \omega)$ lists, first we find the position of the descendant node of x_i in it (the descendant node of x_i doesn't have to exist), then traverse leftwards (rightwards) from the position until the distance overflows, and according to Lemma 3.2, all other nodes in the list needn't be considered.

Each time the distance between two nodes is found to be less equal to ω, these two nodes are linked (by adding cursors point to each other) to build an edge, and the value of distance is recorded as the edge's weight. A weighted undirected graph is obtained, and then a simple graph-partition algorithm is used to get all the cliques. The pseudocode of GC is given in Fig. 3. The graph-partition algorithm is not given here for the sake of space of this paper. We can show that the total cost of finding descendant's position is $h^2 \cdot O(\log n) + h \cdot O(n)$.

It is easy to find that the time complexity of GC not only depends on the total number of keyword nodes, but also strongly relies on the distance threshold. The disadvantage of GC is the uncontrollable efficiency, especially when the distance threshold is large. So, we propose two other algorithms: CC and LCC.

```
ALGORITHM 1 (GRAPH-BASED CLUSTERING)
Input: a hierarchical structure $H$
Output : a set of optimal clusters $C$
1.  for (every list $l$ in $H$)      // top-down
2.    for (every node $x_i$ in $l$)      // left-right
3.      find a rightward neighbor $x_j$
4.      while ($dis(x_i, x_j) <= \omega$)
5.        $link(x_i, x_j)$;      // link two nodes
6.        find next rightward neighbor $x_j$;
7.      for (every list $l'$ in $floor(depth(x_i) \cdot \omega)$ layers below $l$)
8.        $p \leftarrow findDescPosition(x_i, l')$;
9.        traverse leftward and rightward from $p$ until distance
          overflows and link $x_i$ with neighbors close enough;
10. use graph partition algorithm to get optimal cluster set $C$;

findDescPosition($x_i$, $l'$)      // find the position of a descendant of $x_i$ in $l'$
1.  get $x_i$'s descendant $x_j$ in $l'$;
2.  if ($x_i$ is the first element of $l$)
3.    use binary search to find $p$ as the position of $x_j$ in $l'$;
4.  else
5.    search rightwards from $l'.cursor$ to find $p$ as the position of $x_j$;
6.  $l'.cursor \leftarrow p$;
7.  return $p$;
```

Fig. 3. Algorithm GC

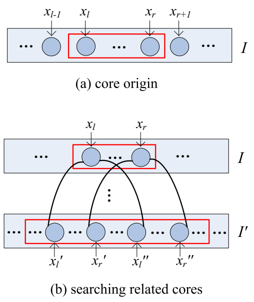


Fig. 4. Illustration of Cores

3.2 CC: Core-Driven Clustering Algorithm for XML Keyword Search

GC is node-driven, which gathers some keyword nodes close enough to each other together. While the idea of algorithm CC is "divide-and-conquer": finds some keyword node sets which are called "cores" in the first place, all the nodes in a core can definitely be clustered together; afterwards, each of which affirmatively contains at least one optimal cluster; finally *optimal clusters* are obtained from these core sets.

Definition 3.2 (Core). Given a distance threshold ω, a keyword node set $R \subseteq X$ is called a core iff the distance between any two nodes in R is less equals to ω.

An optimal cluster is a core, but a core is not necessarily an optimal cluster.

Lemma 3.3. Assume x_l and x_r are any two nodes in the same list of H, and x_l is left of x_r. Given a distance threshold ω and $dis(x_l, x_r) \leq \omega$, then the set of nodes from x_l to x_r (including them) is a core.

Proof: Take any two nodes x_a and x_b from the $(r - l + 1)$ nodes from x_l to x_r, and let x_a be left of x_b. According to Lemma 3.1, $dis(x_a, x_b) \leq dis(x_a, x_r) \leq dis(x_l, x_r) \leq \omega$, so the distance between any two nodes isn't greater than ω. At the beginning of algorithm CC, we traverse H and divide all keyword nodes in H into a number of cores, each of which is called a "core origin".

Definition 3.3 (Core Origin). O_i is a set of certain keyword nodes in H; it is called a core origin iff: (1) O_i is a core; (2) all the nodes in O_i are in the same list I of H; (3) there doesn't exist a core O'_i which satisfies $O_i \subset O'_i$ and all the nodes in O'_i are in I.

A core origin is actually an *optimal cluster* in one list. As illustrated in Fig. 4 (a): assume I is a list in H, x_{l-1}, x_l, x_r and x_{r+1} are four nodes in I and their positions are just as the same as illustrated in the figure, also $dis(x_l, x_r) \leq \omega$, $dis(x_{l-1}, x_l) \geq \omega$, and $dis(x_r, x_{r+1}) \geq \omega$. According to Lemma 3.3, $x_l, \ldots, x_r$ is a core, and easily find: for any node x_a in it, $dis(x_{l-1}, x_a) > dis(_{xl-1}, x_l) > \omega$, and $dis(x_a, x_{r+1}) > dis(x_r, x_{r+1}) > \omega$. On account of Lemma 3.1, the distance between x_a and any node left of x_{l-1} or right of x_{r+1} is greater than ω, consequently $x_l, \ldots, x_r$ is a core origin. It denotes that the positions of a core origin's nodes are continuous in the list.

After getting all the cores origin from H, some cores around each core origin O_i are considered for the purpose of finding optimal clusters which contain all the nodes in O_i. As illustrated in Fig.4 (b), $x_l, \ldots, x_r$ is a core origin, and according to previous discussions, all other nodes in the same list needn't be regarded. Assume I' is a lower list, $x'_l, \ldots, x_l$" is a set of nodes in I' whose distances to x_l are all less equal to ω, and $x'_r, \ldots, x_r$" is a set in I' of nodes whose distances to x_r are all less equal to ω. If the intersection of $x'_l, \ldots, x''_l$ and $x'_r, \ldots, x''_r$ is null, affirmatively there isn't any node in I' could be added into $x'_r, \ldots, x''_r$ to form a core; if the intersection isn't null, it must be $x'_r, \ldots, x''_l$ (it is easy to prove that x'_l can't be right of x'_r and x''_r can't be left of x''_l), therefore we can get a lemma as follows.

Lemma 3.4. If $x'_r, \ldots, x''_l$ is a core, then: (1) the union of $x_l, \ldots, x_r$ and $x'_r, \ldots, x''_l$ is a core; (2) any node in I' other than the nodes of $x'_r, \ldots, x''_l$ cannot be added into $x_l, \ldots, x_r$ to build a core.

Proof: For the first thesis, $x_l, \ldots, x_r$ and $x'_r, \ldots, x''_l$ are both cores, so the only thing needs to be proved is the distance between any two nodes from different sets is less equal to ω. Assume x_a is an arbitrary node in $x_l, \ldots, x_r$, x'_a is an arbitrary node in $x'_r, \ldots, xl''$, and $dis(x_a, x'_a) > \omega$, then according to Lemma 3.2: if $dis(xl, xa') \leq \omega$, the position of x'_a's ancestor in I is left of x_a; if $dis(x_r, x'_a) \leq \omega$, the position of x'_a's ancestor in I is right of x_a. However, $dis(x_l, x'_a)$ and $dis(x_r, x'_a)$ are both less equal to ω, the position of x'_a's ancestor node in I conflicts, so $dis(x_a, x'_a) \leq \omega$. For the second thesis, apparently, the distance between x_r and any node left of x'_r and the distance between x_l and any node left of x''_l all exceed ω.

In the same way, we can prove that, when I' is an upper list Lemma 3.4 still holds. Otherwise, when $x'_r, \ldots, x''_l$ is not a core, we can divide it into several cores, and for each core Lemma 3.4 holds. The similar cores as $x'_r, \ldots, x''_l$ are called "related cores" of $x_l, \ldots, x_r$. After all the related cores of $x_l, \ldots, x_r$ being found, we choose one core from each list except *I*, along with the core origin $x_l, \ldots, x_r$ a core set is obtained, and obviously some optimal clusters containing all the nodes of $x_l, \ldots, x_r$ can be found from it. The pseudocode of algorithm CC is given in Fig. 5, and for simplicity it only considers the case that $x'_r, \ldots, x''_l$ is a core.

Line 1-11 of Algorithm 2 costs $O(n)$; line 6-9 in function *findRelatedCores* costs $O(\log n)$, and line 11 costs $O(n)$, so the total cost of *findRelatedCores* is $O(\log n) + O(n)$. For the function *findOptimalClusters* at line 14 of Algorithm 2, its purpose is to find all optimal clusters which contain all nodes in a cluster origin *O*. Detailed code is omitted here. It is similar to searching cliques in a graph, but the difference is that we consider cores instead of nodes. The core set *S* contains a number of cores and each of which comes from a distinct list; assume there are t cores in *S*, then they have 2^t possible combinations at most; for each of the combination a function similar to *findRelatedCores* needs to be invoked for at most t^2 times; because t is always a small number (less than h), we can see that the complexity of function *findOptimalClusters* is same as *findRelatedCores*.

Also we propose two extreme cases here: (1) when ω is really large, then m is a small number, and the complexity will be $O(n)$; (2) when ω is very small, m tends to n, however line 11 in *findRelatedCores* will cost $O(1)$, so the complexity will be $O(n \cdot \log n)$. Moreover, in the worst case the complexity is $O(n^2)$.

3.3 LCC: Loosened Core-Driven Clustering Algorithm for XML Keyword Search

In most cases users don't have accurate requests for the returned results; we can loosen the restrictions of results for the purpose of improving efficiency. At the beginning of algorithm LCC, *H* is also traversed to get all the cores origin. Afterwards for each list *I* in *H*, two additional lists are built: a head node list *HNL* and a tail node list *TNL*. *HNL* and *TNL* orderly store the first nodes and the last nodes of cores origin in I respectively. Thereafter, for each core origin *O*, we find some cores origins close enough, and then instead of looking for optimal clusters out of them, we easily add all nodes of them into *O* to form a result. The pseudocode of LCC is in Fig. 6. Line 1 of Algorithm 3 is to find all the cores

```
ALGORITHM 2 (CORE-DRIVEN CLUSTERING)
Input: a hierarchical structure $H$
Output : a set of optimal clusters $C$
 1.  for (every list $l$ in $H$)
 2.      $O \leftarrow \emptyset$;
 3.      $x_f \leftarrow$ first node of $l$;
 4.      for (every node $x_i$ in $l$)
 5.          if (dis($x_f$, $x_i$) <= $\omega$)
 6.              add $x_i$ into $O$;
 7.          else
 8.              save $Co$;
 9.              $O \leftarrow \emptyset$;
10.              $x_f \leftarrow x_i$;
11.      save $O$;
12.  for (each core origin $O$)
13.      $S \leftarrow$ findRelatedCores($O$);
14.      $C_i \leftarrow$ findOptimalClusters($S$);
15.      add optimal clusters in $C_i$ into $C$;

findRelatedCores($O$)
 1.  $S \leftarrow \emptyset$;
 2.  for (each layer $l$ except the one $O$ belongs to)
 3.      $R \leftarrow \emptyset$;
 4.      $x_l \leftarrow$ the most left node in $O$;
 5.      $x_r \leftarrow$ the most right node in $O$;
 6.      $p_l \leftarrow$ findAnceOrDescPosition($x_l$, $l$);
 7.      $p_r \leftarrow$ findAnceOrDescPosition($x_r$, $l$);
 8.      use binary search to find $x_l''$;
 9.      use binary search to find $x_r'$;
10.      if ($x_r'$ is $x_l''$ or left of $x_l''$)
11.          add nodes from $x_r'$ to $x_l''$ into $R$ ;
12.          add $R$ into $S$;
```

Fig. 5. Algorithm CC

```
ALGORITHM 3 (LOOSEN CORE-DRIVEN CLUSTERING)
Input: a hierarchical structure $H$
Output : a set of clusters $C$
 1.  find all cores origin;
 2.  for (each core origin $O$)
 3.      $S \leftarrow$ findRelatedCoresOrigin($O$);
 4.      for (each core $R$ in $S$)
 5.          $O \leftarrow O \cup R$;
 6.      add $O$ into $C$;

findRelatedCoresOrigin($O$)
 1.  $S \leftarrow \emptyset$;
 2.  for (each layer $l$ except the one $O$ belongs to)
 3.      $x_l \leftarrow$ the most left node in $O$;
 4.      $x_r \leftarrow$ the most right node in $O$;
 5.      $p_l \leftarrow$ findADPositionInHNL($x_l$, $l$);
 6.      $p_r \leftarrow$ findADPositionInTNL($x_r$, $l$);
 7.      find the most left $x_l''$ satisfy to dis($x_l$, $x_l''$) > $\omega$
         in the nodes of $HNL$ right of $p_l$;
 8.      find the most right $x_r'$ satisfy to dis($x_r$, $x_r'$) > $\omega$
         in the nodes of $HNL$ left of $p_r$;
 9.      if ($x_r'$ is $x_l''$ or left of $x_l''$)
10.          add all cores origin between $x_r'$ and $x_l''$ into $S$;
```

Fig. 6. Algorithm LCC

origin, the processing is the same which in Algorithm 2. For line 4-8 in function *findRelatedCoresOrigin*, the time complexity of each step is $O(\log n)$; and for line 10, the related cores origin only need to be recorded with identifiers rather than be traversed. Assume the number of cores origin is m, then apparently the time complexity of algorithm LCC is $O(n)+O(m\cdot\log n)$; the worst case happens when m tends to n, then it comes to $O(n \cdot logn)$; otherwise, when m is a small number, it is $O(n)$.

4 Ranking of Results

The whole process of ranking mechanism is: firstly sort all the clusters generated by clustering algorithms using the scoring function, and then transform them into MBTs orderly, finally return top-k or all ordered MBTs to users. The first criterion of the scoring function is the number of keywords, obviously containing more keywords indicates a better cluster, and the best clusters are those which contain all the keywords. On the other hand we consider the occurrence frequencies of keywords having no influence on ranking. For those clusters contain the same number of keywords, the criterion of comparison is the average distances of clusters.

Definition 4.1 (Average Distance). The average distance of a cluster C_i is the average value of all the distances between any two nodes in C_i. Assume C_i

contains m nodes, and function $dis_{mean}(C_i)$ is used to get the average distance of C_i, then, $dis_{mean}(C_i) = \frac{\sum_{x_i,x_j \in C_i; x_i \neq x_j} dis(x_i,x_j)}{2C_m^2}, m > 1$

Assume h is the tree height, then the range of $dis_{mean}(C_i)$ is $[1/h, 2h]$. Apparently for a cluster C_i, a smaller $dis_{mean}(C_i)$ indicates that all nodes of C_i are more compact in the tree. Otherwise, if C_i only contains one node, the average distance cannot be defined on it, however we can see that the MBT of it also contains a single node and means almost nothing to users, so this kind of clusters are worst for users. We define function $keywordNum(C_i)$ to get the number of keywords, and $score(C_i)$ as the scoring function of clusters, moreover our final scoring function is as follows:

$$score(C_i) = \begin{cases} h \cdot keywordNum(C_i) + \frac{1}{dis_{mean}(C_i)} & m > 1 \\ 0 & m = 1 \end{cases}$$

Example 4.1: For the four optimal clusters obtained in Example 2.1, their scores are: $score(C_1) = 13.50$, $score(C_2) = 12.50$, $score(C_3) = 18.75$ and $score(C_4) = 19.25$ respectively. So, the order of clusters is: C_4, C_3, C_1, C_2.

5 Experiments

The environment of our experiments is a PC with a 2.8GHZ CPU and 2G RAM; and the software includes: Windows XP, JDK 1.6, and the parser "Xerces". The data sets used to compare three clustering algorithms are DBLP (size 127M, 6332225 nodes) and Treebank (size 82M, 3829511) [12]. We build a vocabulary for each data set, which stores some terms existing in the document; the occurrence frequency of each term is between 5,000 and 15,000. We randomly choose several ones from vocabularies as keywords to search in the documents. The process is repeated for forty times and afterwards the average values are evaluated as the final results.

From Fig. 7 it's easy to find that: (1) given certain keywords and distance threshold, the time costs of three algorithms is ranked as "GC, CC, LCC" (from big to small) except when the distance threshold is very small; (2) with the increasing of the distance threshold, time cost of GC always increases, while time costs of CC and LCC both first increase and then decrease. In Fig. 8 (a), when the distance threshold equals to 0.0, the returned results are those nodes whose labels contain multiple keywords, because each of them is considered as some different nodes; when the distance threshold equals to 6.0, almost all the keyword nodes gather into a few large clusters; also with the threshold increasing, the amount of clusters except single-node ones firstly increases and then reduces to 1. Similar situations happen in other subfigures in Fig 8. We have evaluated the average distances of returned clusters: the results of GC and CC are the same, and LCC's only very litter larger than them (at 4 decimal places). So, the results of GC and CC are the same (all the optimal clusters), and the results of LCC are less and bigger but not quite worse than them. we can see that in any case DBLP has more average keyword nodes, however, Fig. 7 indicates that

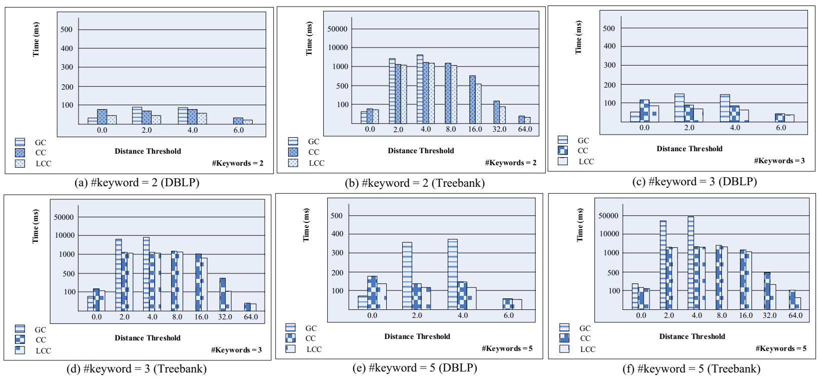


Fig. 7. Efficiencies of Clustering Algorithms

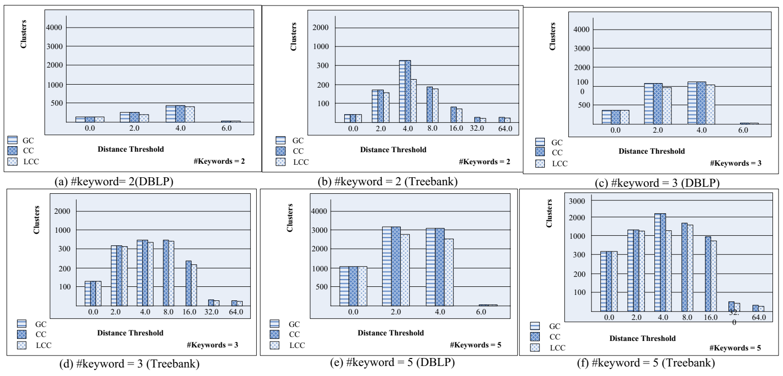


Fig. 8. Result Quantities of Clustering Algorithms

any of the three algorithms costs more time in Treebank than in DBLP with the same conditions, which means the topology of the XML document tree definitely affect the efficiency strongly. The efficiencies become lower when the height is larger and the topology is more complicated.

6 Conclusion

In this paper, to obtain all fragments of the XML document meaningful to users, we propose a novel approach called XKLUSTER, which include a novel semantic distance model, three clustering algorithms (GC, CC, LCC); a ranking mechanism.

Acknowledgments. This research is supported in part by the National Natural Science Foundation of China under grant 60773076, the Key Fundamental Research of Shanghai under Grant 08JC1402500, Xiao-34-1.

References

1. Hristidis, V., Papakonstantinou, Y., Balmin, A.: Keyword proximity search on XML graphs. In: Proceedings of the 19th International Conference on Data Engineering, pp. 367–378. IEEE Computer Society Press, Bangalore (2003)
2. Xu, Y., Papakonstantinou, Y.: Efficient Keyword Search for Smallest LCAs in XML Databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 537–538. ACM, Baltimore (2005)
3. Li, Y., Yu, C., Jagadish, H.V.: Schema-Free XQuery. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, pp. 72–83. Morgan Kaufmann, Toronto (2004)
4. Hristidis, V., Koudas, N., Papakonstantinou, Y., Srivastava, D.: Keyword Proximity Search in XML Trees. IEEE Transactions on Knowledge and Data Engineering 18(4), 525–539 (2006)
5. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: XSEarch: A Semantic Search Engine for XML. In: Proceedings of 29th International Conference on Very Large Data Bases, Berlin, Germany, pp. 45–46 (2003)
6. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: XRANK: Ranked Keyword Search over XML Documents. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, pp. 16–27 (2003)
7. Liu, Z., Chen, Y.: Identifying meaningful return information for XML keyword search. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, pp. 329–340. ACM, Beijing (2007)
8. Kong, L., Gilleron, R., Lemay, A.: Retrieving meaningful relaxed tightest fragments for XML keyword search. In: 12th International Conference on Extending Database Technology, pp. 815–826. ACM, Saint Petersburg (2009)
9. Bao, Z., Ling, T.W., Chen, B., Lu, J.: Effective XML Keyword Search with Relevance Oriented Ranking. In: Proceedings of the 25th International Conference on Data Engineering, pp. 517–528. IEEE, Shanghai (2009)
10. Agrawal, S., Chaudhuri, S., Das, G.: DBXplorer.: A System for Keyword-Based Search over Relational Databases. In: Proceedings of the 18th International Conference on Data Engineering, pp. 5–16. IEEE Computer Society, San Jose (2002)
11. He, H., Wang, H., Yang, J., Yu, P.S.: BLINKS: ranked keyword searches on graphs. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, pp. 305–316. ACM, Beijing (2007)
12. XML Data Repository, `http://www.cs.washington.edu/research/xmldatasets/www/repository.html`

OddBall: Spotting Anomalies in Weighted Graphs

Leman Akoglu, Mary McGlohon, and Christos Faloutsos

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

Abstract. Given a large, weighted graph, how can we find anomalies? Which rules should be violated, before we label a node as an anomaly? We propose the OddBall algorithm, to find such nodes. The contributions are the following: (a) we discover several new rules (power laws) in density, weights, ranks and eigenvalues that seem to govern the so-called "neighborhood sub-graphs" and we show how to use these rules for anomaly detection; (b) we carefully choose features, and design OddBall, so that it is scalable and it can work un-supervised (no user-defined constants) and (c) we report experiments on many real graphs with up to *1.6 million* nodes, where OddBall indeed spots unusual nodes that agree with intuition.

1 Introduction

Given a real graph, with weighted edges, which nodes should we consider as "strange"? Applications of this setting abound: For example, in network intrusion detection, we have computers sending packets to each other, and we want to know which nodes misbehave (e.g., spammers, port-scanners). In a who-calls-whom network, strange behavior may indicate defecting customers, or telemarketers, or even faulty equipment dropping connections too often. In a social network, like FaceBook and LinkedIn, again we want to spot users whose behavior deviates from the usual behavior, such as people adding friends indiscriminately, in "popularity contests".

The list of applications continues: Anomalous behavior could signify irregularities, like credit card fraud, calling card fraud, campaign donation irregularities, accounting inefficiencies or fraud [6], extremely cross-disciplinary authors in an author-paper graph [29], network intrusion detection [28], electronic auction fraud [10], and many others.

In addition to revealing suspicious, illegal and/or dangerous behavior, anomaly detection is useful for spotting rare events, as well as for the thankless, but absolutely vital task of data cleansing [12]. Moreover, anomaly detection is intimately related with the pattern and law discovery: unless the majority of our nodes closely obey a pattern (say, a power law), only then can we confidently consider as outliers the few nodes that deviate.

Most anomaly detection algorithms focus on clouds of multi-dimensional points, as we describe in the survey section. Our goal, on the other hand, is to spot strange nodes in a *graph*, with weighted edges. What patterns and laws do such graphs obey? What features should we extract from each node?

We propose to focus on neighborhoods, that is, a sphere, or a ball (hence the name OddBall) around each node(the *ego*): that is, for each node, we consider the induced sub-graph of its neighboring nodes, which is referred to as the *egonet*. Out of the huge number of numerical features one could extract from the egonet of a given node, we give a carefully chosen list, with features that are effective in revealing outliers. Thus, every node becomes a point in a low-dimensional feature space.

Main contributions of this work are:

1. *Egonet patterns*: We show that egonets obey some surprising patterns (like the *Egonet Density Power Law* (*EDPL*), *EWPL*, *ELWPL*, and *ERPL*), which gives us confidence to declare as outliers the ones that deviate. We support our observations by showing that the *ERPL* yields the *EWPL*.
2. *Scalable algorithm*: Based on those patterns, we propose OddBall, a scalable, un-supervised method for anomalous node detection.
3. *Application on real data*: We apply OddBall[1] to numerous real graphs (DBLP, political donations, and other domains) and we show that it indeed spots nodes that a human would agree are strange and/or extreme.

Of course, there are numerous types of anomalies - we discuss several of them in our technical report [2], but, for brevity, we focus on only the following major types (see Fig.1 for examples and Section 2 for the dataset description):

1. *Near-cliques* and *stars*: Those nodes whose neighbors are very well connected (near-cliques) or not connected (stars) turn out to be "strange": in most social networks, friends of friends are often friends, but either extreme (clique/star) is suspicious.
2. *Heavy vicinities*: If person i has contacted n distinct people in a who-calls-whom network, we would expect that the number of phone calls (weight) would be a function of n. Extreme total weight for a given number of contacts n would be suspicious, indicating, e.g., faulty equipment that forces redialing.
3. *Dominant heavy links*: In the who-calls-whom scenario above, a very heavy single link in the 1-step neighborhood of person i is also suspicious, indicating, e.g., a stalker that keeps on calling only one of his/her contacts an excessive count of times.

The upcoming sections are as follows: We describe the datasets; the proposed method and observed patterns; the experimental results; prior work; and finally the conclusions.

2 Data Description

We studied several unipartite/bipartite, weighted/unweighted large real-world graphs in a variety of domains, described in detail in Table 1. Particularly, unipartite networks include the following: *Postnet* contains post-to-post links in a

[1] Source code of our algorithm can be found at www.cs.cmu.edu/~lakoglu/#tools

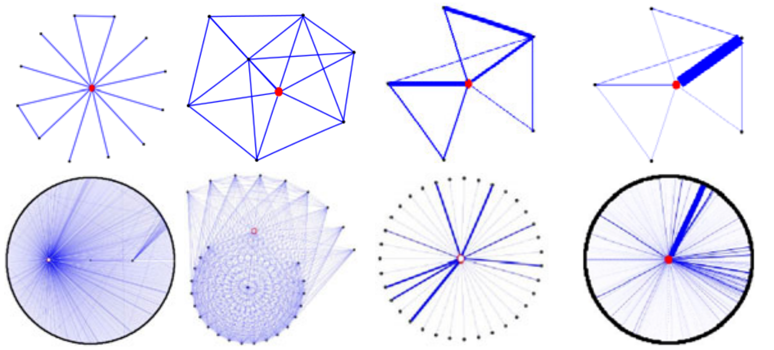


Fig. 1. Types of anomalies that `OddBall` detects. Top row: toy sketches of egonets (ego shown in larger, red circle). Bottom row: actual anomalies spotted in real datasets. (a) A near-star in *Postnet*: `instapundit.com/archives/025235.php`, an extremely long post on Hurricane Katrina relief agencies with numerous links to diverse other posts about donations. (b) A near-clique in *Postnet*: `sizemore.co.uk`, who often linked to its own posts, as well as to its own posts in other blogs. (c) A heavy vicinity in *Postnet*: `blog.searchenginewatch.com/blog` has abnormally high weight w.r.t. the number of edges in its egonet. (d) Dominant edge(s) in *Com2Cand*: In FEC 2004, George W. Bush received a huge donation from a single committee: Democratic National Committee (~\$87M)(!) - in fact, this amount was *spent against* him; next heaviest link (~\$25M): from Republican National Committee.

set of blogs[21], *Enron* contains emails at Enron collected from about 1998 to 2002 (made public by the Federal Energy Regulatory Commission during its investigation), and *Oregon* contains AS peering information inferred from Oregon route-views BGP data. Bipartite networks include the following: *Auth2Conf* contains the publication records of authors to conferences from DBLP, and *Don2Com* and *Com2Cand* are from the U.S. Federal Election Commission in 2004[2], a public record of donations between donors and committees and between committees and political candidates, respectively.

For *Don2Com* and *Com2Cand*, the weights on the edges are actual weights representing donation amounts in dollars. For the remaining weighted datasets, the edge weights are simply the number of occurrences of the edges. For instance, if post i contains k links to another post j, the weight of the edge $e_{i,j}$ is set to k.

In our study, we specifically focused on undirected graphs, but the ideas can easily be generalized to directed graphs.

3 Proposed Method

Borrowing terminology from social network analysis (SNA), "ego" is an individual node.

[2] Parsed dataset from all cycles can be found at `www.cs.cmu.edu/~mmcgloho/fec/data/fec_data.html`

Table 1. Datasets studied in this work

Name	**N**	**E**	**Weights**	**Structure**	**Description**
Postnet	223K	217K	Yes	Unipartite	Network of posts based on citations
Auth2Conf	421K	1M	Yes	Bipartite	DBLP Author/Conference associations
Com2Cand	6K	125K	Yes	Bipartite	2004 US FEC Committee to Candidate donations
Don2Com	1,6M	2M	Yes	Bipartite	2004 US FEC Donor to Committee donations
Enron	36K	183K	No	Unipartite	Email associations at Enron
Oregon	11K	38K	No	Unipartite	AS peering connections

Informally, an ego (=node) of a given network is anomalous if its neighborhood significantly differs from those of others. The basic research questions are: (a) *what features should we use to characterize a neighborhood?* and (b) *what does a 'normal' neighborhood look like?*

Both questions are open-ended, but we give some answers below. First, let's define terminology: the "k-step neighborhood" of node i is the collection of node i, all its k-step-away nodes, and all the connections among all of these nodes – formally, this is the "*induced sub-graph*". In SNA, the 1-step neighborhood of a node is specifically known as its "*egonet*".

How should we choose the value of k steps to study neighborhoods? Given that real-world graphs have small diameter [3], we need to stay with small values of k, and specifically, we recommend k=1. We report our findings only for k=1, because using $k > 1$ does not provide any more intuitive or revealing information, while it has heavy computational overhead, possibly intractable for very large graphs.

3.1 Feature Extraction

The first of our two inter-twined questions is *which statistics/features to extract* from a neighborhood.

There is an infinite set of functions/features that we could use to characterize a neighborhood (number of nodes, one or more eigenvalues, number of triangles, effective radius of the central node, number of neighbors of degree 1, etc etc). Which of all should we use?

Intuitively, we want to select features that (a) are fast-to-compute and (b) will lead us to patterns/laws that most nodes obey, except for a few anomalous nodes. We spend a lot of time experimenting with about a dozen features, trying to see whether the nodes of real graphs obey any patterns with respect to those features (see our technical report [2]). The majority of features lead to no obvious patterns, and thus we do not present them.

The trimmed-down set of features that *are very* successful in spotting patterns, are the following:

1. N_i: number of neighbors (degree) of ego i,
2. E_i: number of edges in egonet i,

3. W_i: total weight of egonet i,
4. $\lambda_{w,i}$: principal eigenvalue of the *weighted* adjacency matrix of egonet i.

The next question is how to look for outliers, in such an n-dimensional feature space, with one point for each node of the graph. In our case, n=4, but one might have more features depending on the application and types of anomalies one wants to detect. A quick answer to this would be to use traditional outlier detection methods for clouds of points using all the features.

In our setting, we can *do better*. As we show next, we group features into carefully chosen pairs, where we show that there are patterns of normal behavior (typically, power-laws). We flag those points that significantly deviate from the discovered patterns as anomalous. Among the numerous pairs of features we studied, the successful pairs and the corresponding type of anomaly are the following:

– E vs N: *CliqueStar*: detects near-cliques and stars
– W vs E: *HeavyVicinity*: detects many recurrences of interactions
– λ_w vs W: *DominantPair*: detects single dominating heavy edge (strongly connected pair)

3.2 Laws and Observations

The second of our research questions is *what do normal neighborhoods look like.* Thus, it is important to find patterns ("laws") for neighborhoods of real graphs, and then report the deviations, if any. In this work, we report some new, surprising patterns:

For a given graph $\mathcal{G}$, node $i \in \mathcal{V}(\mathcal{G})$, and the egonet $\mathcal{G}_i$ of node i;

Observation 1 (*EDPL*: Egonet Density Power Law). *The number of nodes N_i and the number of edges E_i of $\mathcal{G}_i$ follow a power law.*

$$E_i \propto N_i^{\alpha}, \quad 1 \leq \alpha \leq 2.$$

In our experiments the *EDPL* exponent α ranged from 1.10 to 1.66. Fig. 2 illustrates this observation, for several of our datasets. Plots show E_i versus N_i for every node (green points); the black circles are the median values for each bucket of points (separated by vertical dotted lines) after applying logarithmic binning on the x-axis as in [23]; the red line is the least squares(LS) fit on the median points. The plots also show a blue line of slope 2, that corresponds to cliques, and a black line of slope 1, that corresponds to stars. All the plots are in log-log scales.

Observation 2 (*EWPL*: Egonet Weight Power Law). *The total weight W_i and the number of edges E_i of $\mathcal{G}_i$ follow a power law.*

$$W_i \propto E_i^{\beta}, \quad \beta \geq 1.$$

Fig. 3 shows the *EWPL* for (only a subset of) our datasets (due to space limit). In our experiments the *EWPL* exponent β ranged up to 1.29. Values of $\beta > 1$ indicate super-linear growth in the total weight with respect to increasing total edge count in the egonet.

Observation 3 (*ELWPL*: Egonet λ_w Power Law). *The principal eigenvalue $\lambda_{w,i}$ of the weighted adjacency matrix and the total weight W_i of $\mathcal{G}_i$ follow a power law.*

$$\lambda_{w,i} \propto W_i^{\gamma}, \quad 0.5 \leq \gamma \leq 1.$$

Fig. 4 shows the *ELWPL* for a subset of our datasets. In our experiments the *ELWPL* exponent γ ranged from 0.53 to 0.98. γ=0.5 indicates uniform weight distribution whereas γ~1 indicates a dominant heavy edge in the egonet, in which case the weighted eigenvalue closely follows the maximum edge weight. γ=1 if the egonet contains only one edge.

Observation 4 (*ERPL*: Egonet Rank Power Law). *The rank $R_{i,j}$ and the weight $W_{i,j}$ of edge j in $\mathcal{G}_i$ follow a power law.*

$$W_{i,j} \propto R_{i,j}^{\theta}, \quad \theta \leq 0.$$

Here, $R_{i,j}$ is the rank of edge j in the sorted list of edge weights. *ERPL* suggests that edge weights in the egonet have a skewed distribution. This is intuitive; for example in a friendship network, a person could have many not-so-close friends (light links), but only a few close friends (heavy links).

Next we show that if the *ERPL* holds, then the *EWPL* also holds. Given an egonet $\mathcal{G}_i$, the total weight W_i and the number of edges E_i of $\mathcal{G}_i$, let $\mathcal{W}_i$ denote the ordered set of weights of the edges, $W_{i,j}$ denote the weight of edge j, and $R_{i,j}$ denote the rank of weight $W_{i,j}$ in set $\mathcal{W}_i$. Then,

Lemma 1. *ERPL implies EWPL, that is: If $W_{i,j} \propto R_{i,j}^{\theta}$, $\theta \leq 0$, then*

$$W_i \propto E_i^{\beta} \begin{cases} \beta = 1, \text{ if } -1 \leq \theta \leq 0 \\ \beta > 1, \text{ if } \theta < -1 \end{cases}$$

Proof. For brevity, we give the proof for $\theta < -1$ – other cases are similar. If $W_{i,j} = cR_{i,j}^{\theta}$, then $W_{min} = cE_i^{\theta}$ –the least heavy edge l with weight W_{min} is ranked the last, i.e. $R_{i,l} = E_i$. Thus we can write W_i as

$$W_i = W_{min}E_i^{-\theta}\left(\sum_{j=1}^{E_i} j^{\theta}\right) \approx W_{min}E_i^{-\theta}\left(\int_{j=1}^{E_i} j^{\theta} dj\right)$$

$$= W_{min}E_i^{-\theta}\left(\frac{j^{\theta+1}}{\theta+1}\Big|_{j=1}^{E_i}\right) = W_{min}E_i^{-\theta}\left(\frac{1}{-\theta-1} - \frac{1}{(-\theta-1)E_i^{-\theta-1}}\right)$$

For sufficiently large E_i and given $\theta < -1$, the second term in parenthesis goes to 0. Therefore; $W_i \approx c'E_i^{-\theta}$, $c' = \frac{W_{min}}{-\theta-1}$. Since $\theta < -1$, $\beta > 1$. □

3.3 Anomaly Detection

We can easily use the observations given in part 3.2 in anomaly detection since anomalous nodes would behave away from the normal pattern. Let us define the y-value of a node i as y_i and similarly, let x_i denote the x-value of node i for a particular feature pair $f(x, y)$. Given the power law equation $y = Cx^\theta$ for $f(x, y)$, we define the outlierness score of node i to be

$$\textit{out-line(i)} = \frac{max(y_i, Cx_i^\theta)}{min(y_i, Cx_i^\theta)} * log(|y_i - Cx_i^\theta| + 1)$$

Intuitively, the above measure is the “distance to fitting line”. Here we penalize each node with both the *number of times* that y_i deviates from its *expected* value Cx_i^θ given x_i, and with the logarithm of the *amount* of deviation. This way, the minimum outlierness score becomes 0 –for which the actual value y_i is equal to the expected value Cx_i^θ.

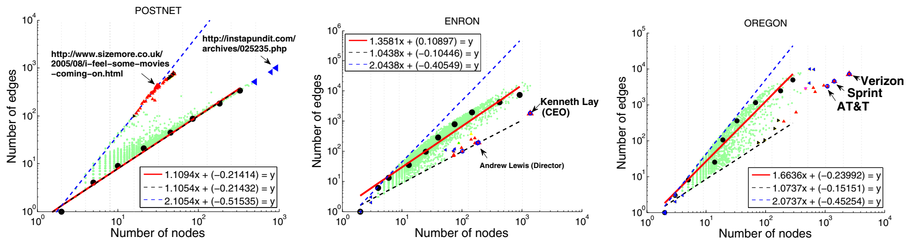
This simple and easy-to-compute method not only helps in detecting outliers, but also provides a way to sort the nodes according to their outlierness scores. However, this method is prone to miss some outliers and therefore could yield false negatives for the following reason: Assume that there exist some points that are far away from the remaining points but that are still located close to the fitting line. In our experiments with real data, we observe that this usually happens for high values of x and y. For example, in Fig. 2(a), the points marked with left-triangles ($\triangleleft$) are almost on the fitting line even though they are far away from the rest of the points.

We want to flag both types of points as outliers, and thus we propose to combine our heuristic with a density-based outlier detection technique. We used LOF [7], which also assigns outlierness scores *out-lof(i)* to data points; but any other outlier detection method would do, as long as it gives such a score. To obtain the final outlierness score of a data point i, one might use several methods such as taking a linear function of both scores and ranking the nodes according to the new score, or merging the two ranked lists of nodes, each sorted on a different score. In our work, we simply used the sum of the two normalized(by dividing by the maximum) scores, that is, *out-score(i)* = *out-line(i)*+*out-lof(i)*.

4 Experimental Results

CliqueStar. Here, we are interested in the communities that the neighbors of a node form. In particular, *CliqueStar* detects anomalies having to do with near-cliques and near-stars. Using *CliqueStar*, we were successful in detecting many anomalies over the unipartite datasets (although it is irrelevant for bipartite graphs since by nature the egonet forms a “star”).

In social media data *Postnet*, we detected posts or blogs that had either all their neighbors connected (cliques) or mostly disconnected (stars). We show some illustrative examples along with descriptions from *Postnet* in Fig. 1. See Fig.2a for the detected outliers on the scatter-plot from the same dataset.



(a) *Postnet* (b) *Enron* (c) *Oregon*

Fig. 2. Illustration of the Egonet Density Power Law (*EDPL*), and the corresponding anomaly *CliqueStar*, with outliers marked with triangles. Edge count versus node count (log-log scale); red line is the LS fit on the median values (black circles); dashed black and blue lines have slopes 1 and 2 respectively, corresponding to stars and cliques. Most striking outlier: Ken Lay (CEO of Enron), with a star-like neighborhood. See Section 5.1.1 for more discussion and Fig.1 for example illustrations from *Postnet*.

In *Enron*(Fig.2b), the node with the highest anomaly score turns out to be "*Kenneth Lay*", who was the CEO and is best known for his role in the Enron scandal in 2001. Our method reveals that none of his over 1K contacts ever sent emails to each other.

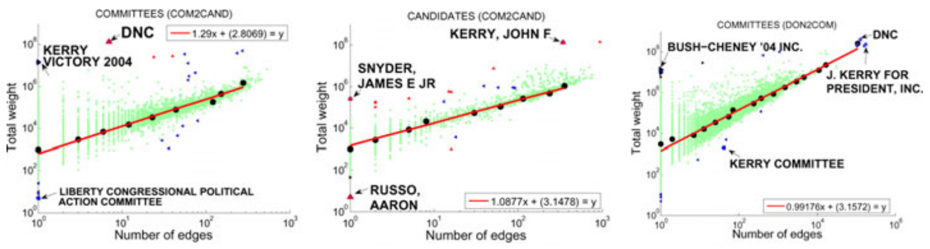
In *Oregon* (Fig.2c), the top outliers are the three large ISPs ("*Verizon*", "*Sprint*" and "*AT&T*").

HeavyVicinity. In our datasets, *HeavyVicinity* detected "heavy egonets", with considerably high total edge weight compared to the number of edges. We mark the anomalies in Fig.3 for several of our datasets. See [2] for results on all the datasets and further discussions.

In *Com2Cand*(Fig.3a), we see that "*Democratic National Committee*" gave away a lot of money compared to the number of candidates that it donated to. In addition, "*(John) Kerry Victory 2004*" donated a large amount to a single candidate, whereas "*Liberty Congressional Political Action Committee*" donated a very small amount ($5), again to a single candidate. Looking at the *Candidates* plot for the same bipartite graph (Fig.3b), we also flagged "*Aaron Russo*", the lone recipient of that PAC. (In fact, Aaron Russo is the founder of the Constitution Party which never ran any candidates, and Russo shut it down after 18 months.)

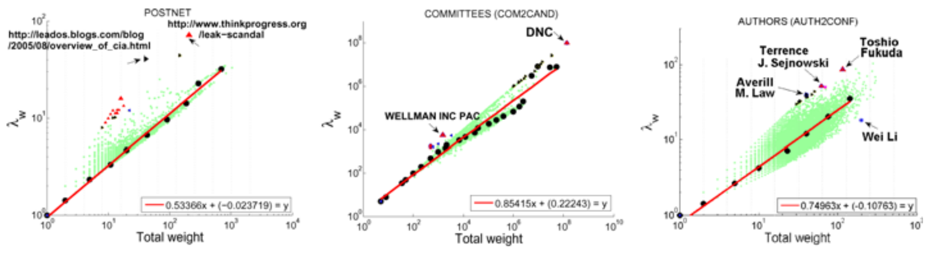
In *Don2Com*(Fig.3c), we see that "*Bush-Cheney '04 Inc.*" received a lot of money from a single donor. On the other hand, we notice that the "*Kerry Committee*" received less money than would be expected looking at the number of checks it received in total. Further analysis shows that most of the edges in its egonet are of weight 0, showing that most of the donations to that committee have actually been returned.

DominantPair. Here, we find out whether there is a single dominant heavy edge in the egonet. In other words, this method detected "bursty" if not exclusive edges.



(a) *Com2Cand* (b) *Com2Cand* (c) *Don2Com*

Fig. 3. Illustration of the Egonet Weight Power Law (*EWPL*) and the weight-edge anomaly *HeavyVicinity*. Plots show total weight vs. total count of edges in the egonet for all nodes (in log-log scales). Detected outliers include Democratic National Committee and John F. Kerry (in FEC campaign donations). See Section 5.2.1 for more discussions.



(a) *Postnet* (b) *Com2Cand* (c) *Auth2Conf*

Fig. 4. Illustration of the Egonet λ_w Power Law (*ELWPL*) and the dominant heavy link anomaly *DominantPair*. Top anomalies are marked with triangles and labeled. See Section 5.2.2 for detailed discussions for each dataset and Fig.1 for an illustrative example from *Com2Cand*.

In *Postnet*(Fig.4a) nodes such as "*ThinkProgress*" 's post on a leak scandal[3] and "*A Freethinker's Paradise*" post[4] linking several times to the "*ThinkProgress*" post were both flagged. On another note, the slope of the fitting line is close to 0.5, pointing to uniform weight distribution in egonets overall. This is expected as most posts link to other posts only once.

In *Com2Cand*(Fig.4b), "*Democratic National Committee*" is one of the top outliers. We would guess that the single large amount of donation was made to "*John F. Kerry*". Counterintuitively, however, we see that that amount was spent for an opposing advertisement against "*George W. Bush*".

DominantPair flagged extremely focused authors (those publish heavily to one conference) in the DBLP data, shown in Fig.3c. For instance, "*Toshio Fukuda*" has 115 papers in 17 conferences (at the time of data collection), with more than half (87) of his papers in one particular conference (ICRA). In addition, "*Averill*

[3] www.thinkprogress.org/leak-scandal
[4] leados.blogs.com/blog/2005/08/overview_of_cia.html

M. Law" has 40 papers published to the "*Winter Simulation Conference*" and nowhere else. On the other extreme, another interesting point is "*Wei Li*", with many papers, who gets them published to as many distinct conferences, probably once or twice to each conference (uniform rather than 'bursty' distribution).

See [2] for results on all the datasets and further discussions.

5 Related Work

5.1 Outlier Detection

Outlier detection has attracted wide interest, being a difficult problem, despite its apparent simplicity. Even the definition of the outlier is hard to give: For instance, Hawkins [16] defines an outlier as "an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism." Similar, but not identical, definitions have been given by Barnett and Lewis [5], and Johnson [19].

Outlier detection methods form two classes, *parametric* (statistical) and *non-parametric* (model-free). The former includes statistical methods that assume prior knowledge of the underlying data distribution [5,16]. The latter class includes *distance-based* and *density-based* data mining methods. These methods typically define as an outlier the (n-D) point that is too far away from the rest, and thus lives in a low-density area [20]. Typical methods include LOF [7] and LOCI [27]. These methods not only flag a point as an outlier but they also give outlierness scores; thus, they can sort the points according to their "strangeness". Many other *density-based* methods especially for large high-dimensional data sets are proposed in [1,4,11,15]. Finally, most clustering algorithms [9,17,25] reveal outliers as a by-product.

5.2 Anomaly Detection in Graph Data

Noble and Cook [26] detect anomalous sub-graphs using variants of the *Minimum Description Length* (MDL) principle. Eberle and Holder [13] use MDL as well as other probabilistic measures to detect several types of anomalies (e.g. unexpected/missing nodes/edges). Frequent subgraph mining [18,30] is used to detect non-crashing bugs in software flow graphs [22]. Chakrabarti [8] uses MDL to spot anomalous edges. Sun et al. [29] use proximity and random walks, to assess the normality of nodes in bipartite graphs. OutRank and LOADED [14,24] use similarity graphs of objects to detect outliers.

In contrast to the above, we work with *unlabeled* graphs. We explicitly focus on nodes, while interactions are also considered implicitly as we study *neighborhood sub-graphs*. Finally, we consider both bipartite and unipartite graphs as well as edge *weights*.

6 Conclusion

This is one of the few papers that focus on anomaly detection in graph data, including weighted graphs. We propose to use "egonets", that is, the induced sub-graph of the node of interest and its neighbors; and we give a small, carefully designed list of numerical features for egonets. The major contributions are the following:

1. Discovery of new patterns that egonets follow, such as patterns in density (Obs.1: *EDPL*), weights (Obs.2: *EWPL*), principal eigenvalues (Obs.3: *ELWPL*), and ranks (Obs.4: *ERPL*). Proof of Lemma 1, linking the *ERPL* to the *EWPL*.
2. `OddBall`, a fast, un-supervised method to detect abnormal nodes in weighted graphs. Our method does not require any user-defined constants. It also assigns an "outlierness" *score* to each node.
3. Experiments on real graphs of over 1M nodes, where `OddBall` reveals nodes that indeed have strange or extreme behavior.

Future work could generalize `OddBall` to time-evolving graphs, where the challenge is to find patterns that neighborhood sub-graphs follow and to extract features incrementally over time.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grants No. IIS0808661, iCAST, and under the auspices of the U.S. Department of Energy by UC Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. This work is also partially supported by an IBM Faculty Award, a SPRINT gift, with additional funding from Intel, and Hewlett-Packard. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

References

1. Aggarwal, C.C., Yu, P.S.: Outlier detection for high dimensional data. In: SIGMOD, pp. 37–46 (2001)
2. Akoglu, L., McGlohon, M., Faloutsos, C.: Anomaly detection in large graphs. CMU-CS-09-173 Technical Report (2009)
3. Albert, R., Jeong, H., Barabasi, A.-L.: Diameter of the world wide web. Nature (401), 130–131 (1999)
4. Arning, A., Agrawal, R., Raghavan, P.: A linear method for deviation detection in large databases. In: KDD, pp. 164–169 (1996)
5. Barnett, V., Lewis, T.: Outliers in Statistical Data. John Wiley and Sons, Chichester (1994)
6. Bay, S., Kumaraswamy, K., Anderle, M.G., Kumar, R., Steier, D.M.: Large scale detection of irregularities in accounting data. In: ICDM (2006)
7. Breunig, M.M., Kriegel, H.-P., Ng, R.T., Sander, J.: Lof: Identifying density-based local outliers. In: SIGMOD, pp. 93–104 (2000)
8. Chakrabarti, D.: AutoPart: Parameter-free graph partitioning and outlier detection. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 112–124. Springer, Heidelberg (2004)

9. Chaoji, V., Hasan, M.A., Salem, S., Zaki, M.J.: Sparcl: Efficient and effective shape-based clustering. In: ICDM (2008)
10. Chau, D.H., Pandit, S., Faloutsos, C.: Detecting fraudulent personalities in networks of online auctioneers. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 103–114. Springer, Heidelberg (2006)
11. Chaudhary, A., Szalay, A.S., Moore, A.W.: Very fast outlier detection in large multidimensional data sets. In: DMKD (2002)
12. Dasu, T., Johnson, T.: Exploratory Data Mining and Data Cleaning. Wiley Interscience, Hoboken (May 2003)
13. Eberle, W., Holder, L.B.: Discovering structural anomalies in graph-based data. In: ICDM Workshops, pp. 393–398 (2007)
14. Ghoting, A., Otey, M.E., Parthasarathy, S.: Loaded: Link-based outlier and anomaly detection in evolving data sets. In: ICDM (2004)
15. Ghoting, A., Parthasarathy, S., Otey, M.E.: Fast mining of distance-based outliers in high-dimensional datasets. Data Min. Knowl. Discov. 16(3), 349–364 (2008)
16. Hawkins, D.: Identification of outliers. Chapman and Hall, Boca Raton (1980)
17. Hu, T., Sung, S.Y.: Detecting pattern-based outliers. Pattern Recognition Letters 24(16) (2003)
18. Jin, R., Wang, C., Polshakov, D., Parthasarathy, S., Agrawal, G.: Discovering frequent topological structures from graph datasets. In: KDD (2005)
19. Johnson, R.A., Wichern, D.W.: Applied Multivariate Statistical Analysis. Prentice Hall, Englewood Cliffs (1998)
20. Knorr, E.M., Ng, R.T.: Algorithms for mining distance-based outliers in large datasets. In: VLDB, pp. 392–403 (1998)
21. Leskovec, J., McGlohon, M., Faloutsos, C., Glance, N., Hurst, M.: Cascading behavior in large blog graphs: Patterns and a model. In: Society of Applied and Industrial Mathematics: Data Mining (2007)
22. Liu, C., Yan, X., Yu, H., Han, J., Yu, P.S.: Mining behavior graphs for "backtrace" of noncrashing bugs. In: SDM (2005)
23. McGlohon, M., Akoglu, L., Faloutsos, C.: Weighted graphs and disconnected components: Patterns and a model. In: ACM SIGKDD (2008)
24. Moonesinghe, H.D.K., Tan, P.-N.: Outrank: a graph-based outlier detection framework using random walk. International Journal on Artificial Intelligence Tools 17(1) (2008)
25. Ng, R.T., Han, J.: Efficient and effective clustering methods for spatial data mining. In: VLDB, pp. 144–155 (1994)
26. Noble, C.C., Cook, D.J.: Graph-based anomaly detection. In: KDD, pp. 631–636 (2003)
27. Papadimitriou, S., Kitagawa, H., Gibbons, P.B., Faloutsos, C.: Loci: Fast outlier detection using the local correlation integral. In: ICDE (2003)
28. Sequeira, K., Zaki, M.J.: Admit: anomaly-based data mining for intrusions. In: KDD (2002)
29. Sun, J., Qu, H., Chakrabarti, D., Faloutsos, C.: Neighborhood formation and anomaly detection in bipartite graphs. In: ICDM (2005)
30. Yan, X., Han, J.: gspan: Graph-based substructure pattern mining. In: ICDM (2002)

Robust Outlier Detection Using Commute Time and Eigenspace Embedding

Nguyen Lu Dang Khoa and Sanjay Chawla

School of Information Technologies, University of Sydney
Sydney NSW 2006, Australia
khoa@it.usyd.edu.au, sanjay.chawla@sydney.edu.au

Abstract. We present a method to find outliers using 'commute distance' computed from a random walk on graph. Unlike Euclidean distance, commute distance between two nodes captures both the distance between them and their local neighborhood densities. Indeed commute distance is the Euclidean distance in the space spanned by eigenvectors of the graph Laplacian matrix. We show by analysis and experiments that using this measure, we can capture both global and local outliers effectively with just a distance based method. Moreover, the method can detect outlying clusters which other traditional methods often fail to capture and also shows a high resistance to noise than local outlier detection method. Moreover, to avoid the $O(n^3)$ direct computation of commute distance, a graph component sampling and an eigenspace approximation combined with pruning technique reduce the time to $O(nlogn)$ while preserving the outlier ranking.

Keywords: outlier detection, commute distance, eigenspace embedding, random walk, nearest neighbor graph.

1 Introduction

Unlike other data mining techniques which extract common or frequent patterns, the focus of outlier detection is on finding abnormal or rare observations in the data. Standard techniques for outlier detection include statistical [7,14], distance based [2,10] and density based [3] approaches. However, standard statistical and distance based approaches can only find global outliers which are extremes with respect to all observations in the dataset. On the other hand local outliers are extremes with respect to their neighborhood observations, but may not be extremes with respect to all other observations in the dataset [16]. A well-known method for detecting local outliers is the Local Outlier Factor (LOF), which is a density based approach [3]. The downside of LOF is the outlier score of each observation only considers its local neighborhood and does not have the global view over all the dataset. Recently, Moonesinghe and Tan [13] proposed a method called OutRank to detect outlier using a random walk on graph. The outlier score is the connectivity of each node which is computed from a stationary random walk. This method cannot find outlying clusters where the node

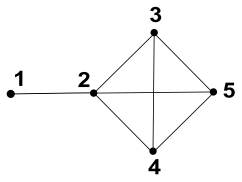


Fig. 1. Example of CD. Edge e_{12} has a larger CD than all edges in the cluster while its Euclidean distance is the same or smaller than their Euclidean distances.

connectivities are still high. An excellent survey by Chandola et. al [4] provides a more detailed view on outlier detection techniques.

In this paper, we present a new method to find outliers using a measure called commute time distance, or commute distance for short (CD)[1]. CD is a well-known measure derived from a random walk on graph [11]. The CD between two nodes i and j in the graph is the number of steps that a random walk, starting from i will take to visit j and then come back to i for the first time. Indeed CD is a Mahalanobis distance in the space spanned by eigenvectors of the graph Laplacian matrix. Unlike traditional Mahalanobis distance, CD between two nodes can capture both the distance between them and their local neighborhood densities so that we can capture both global and local outliers using distance based methods such as methods in [2,10]. Moreover, the method can be applied directly to graph data.

To illustrate, consider a graph of five data points shown in Figure 1, which is built from a dataset of five observations. Denote $d_{ED}(i,j)$ and $d_{CD}(i,j)$ as an Euclidean distance and a CD between observations i and j, respectively. The distances between all pairs of observations are in Table 1.

Table 1. The Euclidean distance and CD for the graph in Figure 1

	Euclidean Distance					Commute Distance				
Index	1	2	3	4	5	1	2	3	4	5
1	0	**1.00**	1.85	1.85	2.41	0	**12.83**	19.79	19.79	20.34
2	1.00	0	1.00	1.00	1.41	12.83	0	6.96	6.96	7.51
3	1.85	1.00	0	1.41	1.00	19.79	6.96	0	7.51	6.96
4	1.85	1.00	1.41	0	1.00	19.79	6.96	7.51	0	6.96
5	2.41	1.41	1.00	1.00	0	20.34	7.51	6.96	6.96	0

It can be seen that $d_{CD}(1,2)$ is much larger than $d_{CD}(i,j)$ ($(i,j) \in \{2,3,4,5\}$, $i \neq j$) even though $d_{ED}(i,j)$ have the same or larger Euclidean distances than $d_{ED}(1,2)$. The CD from an observation outside the cluster to an observation inside the cluster is significantly larger than the CDs of observations inside the cluster. Since CD is a metric, a distance based method can be used to realize that point 1 is far away from other points using CD. Therefore, the use of CD is promising for identifying outliers. The contributions of this paper are as follows:

[1] A preliminary version of this work appeared as a technical report [8].

– We prove that CD can naturally capture the local neighborhood density and establish a relationship between CD and local outlier detection.
– We propose an outlier detection method using the CD metric to detect global and local outliers. The method can also detect outlying clusters which traditional methods often fail to capture. Moreover, the method is shown to be more resistant to noise than other local outlier detection methods.
– We accelerate the computation of CD using a graph component sampling and an eigenspace approximation to avoid $O(n^3)$ computation. Furthermore, pruning technique is used to calculate the CD 'on demand'. All of them speed up the method significantly to $O(nlogn)$ while preserving the outlier ranking.

The remainder of the paper is organized as follows. Section 2 reviews the theory of random walk on graph and CD. In Section 3, we introduce the method to detect outliers with the CD measure. Section 4 presents a way to approximate CD and accelerate the algorithm. In Section 5, we evaluate our approach using experiments on real and synthetic datasets. Section 6 is the conclusion.

2 Background

2.1 Random Walk on Graph and Stationary Distribution

The random walk on a graph is a sequence of nodes described by a finite Markov chain which is time-reversible [11]. The probability that the random walk on node i at time t selects node j at time $t+1$ is determined by the edge weight on the graph: $p_{ij} = P(s(t+1) = j|s(t) = i) = w_{ij}/d_{ii}$ where $d_{ii} = \sum_{j \in adj(i)} w_{ij}$ and $adj(i)$ is a set of neighbors of node i.

Let P be the transition matrix with entry p_{ij}, A is the graph adjacency matrix, and D is the diagonal matrix with entries d_{ii}. Then $P = D^{-1}A$. Denote $\pi_i(t)$ as the probability of reaching node i at time t, $\pi(t) = [\pi_1(t), \pi_2(t), ..., \pi_n(t)]^{\mathrm{T}}$ as the state probability distribution at time t, the state on transforming is $\pi(t+1) = P^{\mathrm{T}}\pi(t)$ and thus $\pi(t) = (P^{\mathrm{T}})^t\pi(0)$ where $\pi(0)$ is an initial state distribution. The distribution $\pi(t)$ is stationary if $\pi(t) = \pi(0)$ for all $t > 0$.

2.2 Commute Distance

This section reviews two measures of a random walk called hitting time $h(i,j)$ and commute time $c(i,j)$ [11]. The hitting time $h(i,j)$ is the expected number of steps a random walk starting at i will take to reach j for the first time:

$$h(i,j) = \begin{cases} 0 & \text{if } i = j \\ 1 + \sum_{k \in adj(i)} p_{ik} h(k,j) & \text{otherwise.} \end{cases} \quad (1)$$

The commute time, which is known to be a metric and that is the reason for the term 'commute distance' [6], is the expected number of steps that a random walk starting at i will take to reach j once and go back to i for the first time:

$$c(i,j) = h(i,j) + h(j,i). \quad (2)$$

The CD can be computed from the Moore-Penrose pseudoinverse of the graph Laplacian matrix [9,6]. Denote $L = D - A$ and L^+ as the graph Laplacian matrix and its pseudoinverse respectively, the CD is:

$$c(i,j) = V_G(l^+_{ii} + l^+_{jj} - 2l^+_{ij}), \tag{3}$$

where $V_G = \sum_{i=1}^{n} d_{ii}$ is the volume of the graph and l^+_{ij} is the (i,j) element of L^+. Equation 3 can be written as

$$c(i,j) = V_G(e_i - e_j)^{\mathrm{T}} L^+ (e_i - e_j), \tag{4}$$

where e_i is the i-th column of an $(n \times n)$ identity matrix I [15]. Consequently, $c(i,j)^{1/2}$ is a distance in the Euclidean space spanned by the e_i's.

3 Commute Distance Based Outlier Detection

3.1 A Proof of Commute Distance Property for Outlier Detection

We now show that CD is a good metric for local outlier detection.

Lemma 1. *The expected number of steps that a random walk which has just visited node i will take before returning back to i is V_G/d_{ii}.*

Proof. For the proof of this Lemma, see [11].

Theorem 1. *Given a cluster C and a point s outside C connected to a point t on the boundary of C (Fig. 2a). If C becomes denser (by adding more points or edges), the CD between s and t increases.*

Proof. From Lemma 1, the expected number of steps that a random walk which has just visited node s will take before returning back to s is $V_G/d_{ss} = V_G/w_{st}$. Since the random walk can only move from s to t, $V_G/w_{st} = h(s,t) + h(t,s) = c(s,t)$ (Fig. 2b). If cluster C becomes denser, there are more edges in cluster C. As a result, V_G increases while w_{st} is unchanged. So the CD between s and t (i.e $c(s,t)$) increases. □

As shown in Theorem 1, the denser the cluster, the larger the CD between a point s outside the cluster to a point t in the cluster. That is the reason why we can effectively detect local outliers using CD.

3.2 Outlier Detection Using Commute Distance

This section introduces a method based on CD to detect outliers. As CD is a metric and captures both the distance between nodes and their local neighborhood densities, we can use a CD based method to find global and local outliers.

First, a mutual k_1-nearest neighbor graph is constructed from the dataset. The mutual k_1-nearest neighbor graph connects nodes u and v if u belongs to the k_1 nearest neighbors of v and v belongs to the k_1 nearest neighbors of u. The

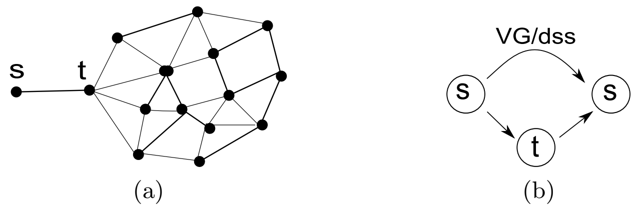


Fig. 2. The CD from an outlier to an observation in a cluster increases when the cluster is denser

reason for choosing mutual k_1-nearest neighbor graph is that this graph tends to connect nodes within cluster of similar densities, but does not connect nodes from clusters of different densities [12]. Therefore, outliers are isolated and data clusters form graph components in mutual k_1-nearest neighbor graph. Moreover, the mutual k_1-nearest neighbor graph with n nodes ($k_1 \ll n$) is usually sparse, which has an advantage in computation. If the data has coordinates, we can use kd-tree to avoid $O(n^2)$ searching of nearest neighbors. The edge weights are inversely proportional to their Euclidean distances. However, it is possible that the mutual k_1-nearest neighbor graph is not connected so that we cannot apply random walk on the whole graph. One approach to make the graph connected is to find its minimum spanning tree and add the edges of the tree to the graph.

Then the graph Laplacian matrix L and its pseudoinverse L^+ are computed. After that the pairwise CDs between any two observations are calculated from L^+. Finally, the distance based outlier detection using CD with pruning technique proposed by Bay and Schwabacher [2] is used to find the top N outliers. The main idea of pruning is that an observation is not an outlier if its average distance to k_2 current nearest neighbors is less than the score of the weakest outlier among top N found so far. Using this approach, a large number of non-outliers can be pruned without carrying out a full database scan. The outlier score used is the average distance of an observation to its k_2 nearest neighbors. Suitable values for k_1 (for building the nearest neighbor graph) and k_2 (for estimating the outlier score) will be presented in the experiments.

4 Graph Component Sampling and Eigenspace Approximation

While CD is a robust measure for detecting both global and local outliers, its main drawback is its computational time. The direct computation of CD from L^+ is proportional to $O(n^3)$ which is not feasible for large graphs (n is the number of nodes). In this work, the graph components are sampled to reduce the graph size and then eigenspace approximation in [15] is applied to approximate the CD on the sampled graph.

4.1 Graph Sampling

An easy way to sample a graph is selecting nodes from it uniformly at random. However, sampling in this way can break the graph geometry structure and outliers may not be chosen in sampling. To resolve this, we propose a sampling strategy called component sampling. After creating the mutual k_1-nearest neighbor graph, the graph tends to have many connected components corresponding to different data clusters. Outliers are either isolated nodes or nodes in very small components. For nodes in normal components (we have a threshold to distinguish between normal and outlying components), they are uniformly sampled with the same ratio $p = 50k_1/n$, which is chosen from experimental results. For nodes in outlying components, we sample all of them. Then we rebuild a mutual k_1-nearest neighbor graph for the sampled data. Sampling in this way will maintain the geometry of the original graph and the relative densities of normal clusters. Outliers are also not sampled in this sampling strategy.

4.2 Eigenspace Approximation

Because the Laplacian matrix L $(n \times n)$ is symmetric and has rank $n-1$ [5], it can be decomposed as $L = VSV^{\mathrm{T}}$, where V is the matrix containing eigenvectors of L as columns and S is the diagonal matrix with the corresponding eigenvalues $\lambda_1 = 0 < \lambda_2 < ... < \lambda_n$ on the diagonal. Then $L^+ = VS^+V^{\mathrm{T}}$ where S^+ is the diagonal matrix with entries $\lambda_1^+ = 1/\lambda_2 > \lambda_2^+ = 1/\lambda_3 > ... > \lambda_{n-1}^+ = 1/\lambda_n > \lambda_n^+ = 0$. Equation 4 can be written as $c(i,j) = V_G(x_i - x_j)^{\mathrm{T}}(x_i - x_j)$ where $x_i = S^{+1/2}V^{\mathrm{T}}e_i$ [15]. Therefore, the CD between nodes on the graph can be viewed as the Euclidean distance in the space spanned by eigenvectors of the graph Laplacian matrix.

Denote $\tilde{V}$, $\tilde{S}$ as a matrix containing m largest eigenvectors of L^+ and its corresponding diagonal matrix, and $\tilde{x}_i = \tilde{S}^{+1/2}\tilde{V}^{\mathrm{T}}e_i$, the approximate CD is

$$\tilde{c}(i,j) = V_G(\tilde{x}_i - \tilde{x}_j)^{\mathrm{T}}(\tilde{x}_i - \tilde{x}_j), \tag{5}$$

The CD $c(i,j)$ in an n dimensional space is transformed to the CD $\tilde{c}(i,j)$ in an m dimensional space. Therefore, we just need to compute the m smallest eigenvectors with nonzero eigenvalues of L (i.e the largest eigenvectors of L^+) to approximate the CD. This approximation is bounded by $\|c(i,j) - \tilde{c}(i,j)\| \leq V_G \sum_{i=1}^{m} \lambda_i^+$ [15].

4.3 Algorithm

The proposed method is outlined in Algorithm 1. We create the sampled graph from the data using graph components sampling. Then the graph Laplacian L of the sampled graph and matrix $\tilde{V}$ (m smallest eigenvectors with nonzero eigenvalues of L) are computed. Since we use the pruning technique, we do not need to compute the approximate CD for all pairs of points. Instead, we compute it 'on demand' using the formula in equation 3 where $\tilde{l}_{ij}^+ = \sum_{k=1}^{m} \lambda_k^+ v_{ik} v_{jk}$, v_{jk} and v_{jk} are entries of matrix $\tilde{V}$.

4.4 The Complexity of the Algorithm

The k-nearest neighbor graph with n nodes is built in $O(nlogn)$ using kd-tree with the assumption that the dimensionality of the data is not very high. The average degree of each node is $O(k)$ ($k \ll n$). So the graph is sparse and thus finding connected components take $O(kn)$. After sampling, the size of graph is $O(n_s)$ ($n_s \ll n$). The standard method for eigen decomposition of L is $O(n_s^3)$. Since L is sparse, it would take $O(Nn_s) = O(kn_s^2)$ where N is the number of nonzeros. The computation of just the m smallest eigenvectors ($m < n_s$) is less expensive than that.

The typical distance based outlier detection takes $O(n_s^2)$ for the neighborhood search. Pruning can scale it nearly linear. We only need to compute the CDs $O(n_s)$ times each of which takes $O(m)$.

So the time needed for two steps is proportional to $O(nlogn + kn + kn_s^2 + mn_s) = O(nlogn)$ as $n_s \ll n$.

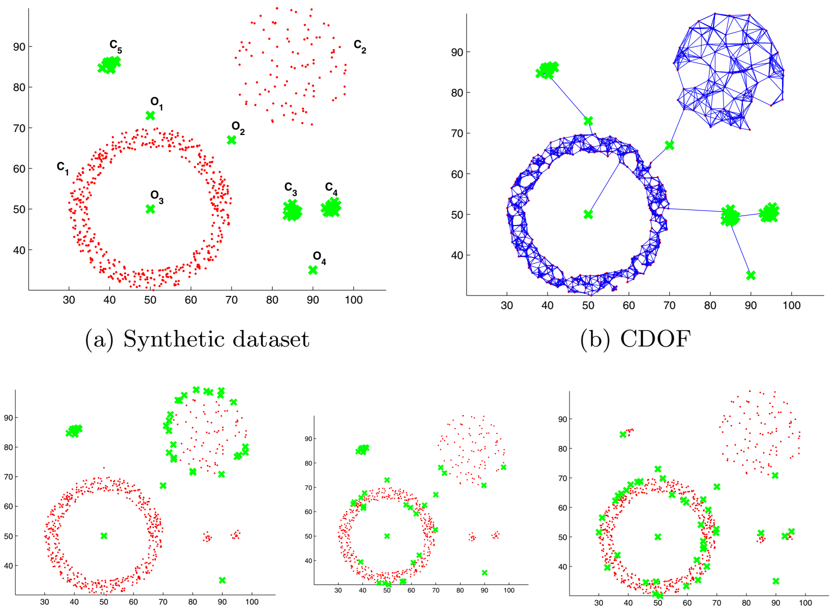
Algorithm 1. Commute Distance Based Outlier Detection with Graph Component Sampling and Eigenspace Approximation.

Input: Data matrix X, the numbers of nearest neighbors k_1 and k_2, the numbers of outliers to return N
Output: Top N outliers

```
1: Construct the mutual $k_1$-nearest neighbor graph from the dataset
2: Do the graph component sampling
3: Reconstruct the mutual $k_1$-nearest neighbor graph from sampled data
4: Compute the Laplacian matrix of the sampled graph and its $m$ smallest eigenvectors
5: Find top $N$ outliers using the CD based technique with pruning rule (using $k_2$)
6: Return top $N$ outliers
```

5 Experiments and Analysis

In this section, the effectiveness of CD as a measure for outlier detection is evaluated. Firstly, the ability of the distance based technique using CD (denoted as CDOF) in finding global, local outliers, and outlying clusters was tested in a synthetic dataset. The distance based technique using Euclidean distance [2] (denoted as EDOF), LOF [3], and OutRank [13] (denoted as ROF and the same graph of CDOF was used) were also used to compare with CDOF. Secondly, the effectiveness of CDOF was evaluated in a real dataset. Thirdly, we have shown that CDOF is more resistant to small perturbations to data than LOF. Finally the performance and effectiveness of approximate CDOF were evaluated. The experiments were conducted on a workstation with an 3GHz Intel Core2 Duo processor and 2GB of main memory in Windows XP.



(a) Synthetic dataset (b) CDOF

(c) EDOF (d) LOF (e) ROF

Fig. 3. Comparison of the results of EDOF, LOF, ROF, and CDOF. CDOF can detect all global, local outliers, and outlying clusters effectively.

5.1 Synthetic Dataset

Figure 3a shows a 2-dimensional synthetic dataset. It contains one dense cluster of 500 observations (C_1) and one sparse cluster of 100 observations (C_2). Moreover, there are three small outlying clusters with 12 observations each (C_{3-5}) and four outliers (O_{1-4}). All the clusters were generated from a Normal distribution. O_2, O_3, O_4 are global outliers which are far away from other clusters. O_1 is a local outlier of dense cluster C_1.

In the following experiments for this dataset, the numbers of nearest neighbors are $k_1 = 10$ (for building the graph), $k_2 = 15$ (for estimating the outlier score. Since the size of outlying clusters is twelve, fifteen is a reasonable number to estimate the outlier scores), and the number of top outliers is $N = 40$ (the total observations in three outlying clusters and four outliers). The results are shown in Figure 3. The 'x' signs mark the top outliers found by each method. The figure shows that EDOF cannot detect local outlier O_1. Both EDOF and LOF cannot find two outlying cluster C_3 and C_4. The reason is those two clusters are near each other with similar densities and consequently for each point in the two clusters the average distance to its nearest neighbors is small and the relative density is similar to that of its neighbors. Moreover, ROF outlier score is actually the node probability distribution when the random walk is stationary

[13]. Therefore, it is d_{ii}/V_G [11], which is small for outliers[2]. Therefore, it cannot capture nodes in the outlying clusters where d_{ii} is large. For degree one outlying nodes, ROF and CDOF have similar scores. The result in Figure 3b shows that CDOF can identify all the outliers and outlying clusters. The key point is in CD, inter-cluster distance is significantly larger then intra-cluster distance even if the two clusters are near in the Euclidean distance.

5.2 Real Dataset

In this experiment, CDOF was used to find outliers in an NBA dataset. The dataset contains information of all the players in the famous basketball league in the US in year 1997-1998. There were 547 players and six attributes were used: position, points per game, rebounds per game, assists per game, steals per game and blocks per game. Point and assist reflect the offensive ability of a player while steal and block show how good a player is in defending. Rebound can be either offensive or defensive but total rebound is usually an indicator of defensive ability. The results are shown in Table 2 with the ranking and statistics of top five outliers. The table also shows the maximums, averages, and standard deviations for each attribute over all players.

Table 2. The outlying NBA players

Rank	Player	Position	Points	Rebounds	Assists	Steals	Blocks
1	Dikembe Mutombo	Center	13.43	11.37	1.00	0.41	3.38
2	Dennis Rodman	Forward	4.69	15.01	2.88	0.59	0.23
3	Michael Jordan	Guard	28.74	5.79	3.45	1.72	0.55
4	Shaquille O'neal	Center	28.32	11.35	2.37	0.65	2.40
5	Jayson Williams	Forward	12.88	13.58	1.03	0.69	0.75
	Max		28.74	15.01	10.54	2.59	3.65
	Average		7.69	3.39	1.78	0.70	0.40
	Standard deviation		5.65	2.55	1.77	0.48	0.53

Dikembe Mutombo was ranked as the top outlier. He had the second highest blocks (5.6 times of standard deviation away from mean), the highest rebounds for center players, and high points. It is rare to have good scores in three or more different statistics and he was one of the most productive players. Dennis Rodman and Michael Jordan took the second and third positions because of their highest scores in rebound and point (4.6 and 3.7 times of standard deviation away from mean, respectively). Dennis Rodman was a rare case because his points were quite low among high rebound players as well. The next was Shaquille O'Neal who had the second highest points and high rebounds. He was actually the best scoring center and is likely a local outlier among center players. Finally, Jayson Williams had the second highest rebounds. It is interesting to note that except for Dennis Rodman because of his bad behaviour in the league, the other four players were listed in that year as members of All-Stars team [1].

[2] This score has not been explicitly stated in the ROF paper.

5.3 Sensitivity to Data Perturbation

In this section LOF and CDOF were compared on their ability to handle 'noise' perturbations in data. Recall that $LOF(p)$ is the ratio between the average relative density of the nearest neighbors q of p over the relative density of p. $LOF(p)$ is high (i.e p is outlier) if p's neighborhood area is sparse and q's neighborhood area is dense. Suppose that noise is uniformly distributed in the data space, it is obvious that the noise will have more effect on outliers than points in clusters. The noise data can be neighbors of outliers and their neighborhood are also sparse. Thus the numerator in $LOF(p)$ formula where p is outlier reduces considerably while the denominator increases. As a result, LOFs of outliers may reduce significantly but they does not change much for points inside the clusters. Therefore, the relative rankings of data may not be preserved. On the other hand, uniform noise changes the nearest neighbor graph for CDOF in the way that degrees of outliers will increase but are still much smaller than degree of points inside the clusters. Thus there will still be a big difference between inter-cluster and intra-cluster CDs. That maintains the higher scores for outliers than the points inside the cluster.

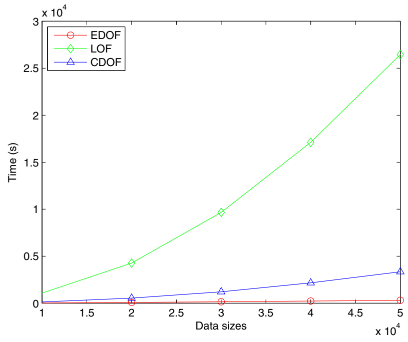
To show this, we randomly added 10% noise from a uniform distribution to the synthetic dataset in Section 5.1 and applied LOF and CDOF in the new dataset. Then noise data was removed from the ranking. Two criteria were used to compare two methods: Spearman rank test for the ranking of the whole dataset and the similarity between the sets of top outliers before and after adding noise. The results were averaged over ten trials. Spearman rank test in LOF was 0.01 while the it was 0.48 for CDOF. It shows the relative ranking by LOF changes significantly due to noise effect. After adding noise, there were 62% of the original outliers still in the top outlier list for LOF while it was 92% for CDOF. CDOF is less sensitive as it combines local and global views of the data.

Since noise is a kind of outlier, we cannot distinguish between outliers and noise but the preliminary results for noise effect show that the proposed method is more resistance to noise than the local outlier detection method.

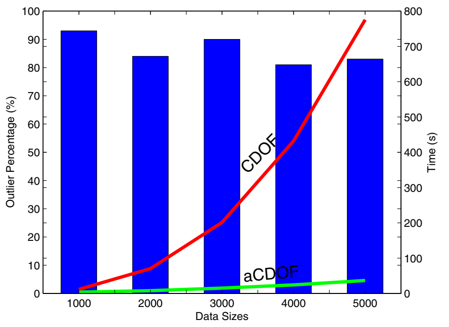
5.4 Performances of the Proposed Method

In the following experiments, we compared the performances of EDOF, LOF, and approximate CDOF mentioned in Section 4. The experiment was performed using five synthetic datasets, each of which contained different clusters generated from Normal distributions and a number of random points. The number of clusters, the sizes, and the locations of the clusters were also chosen randomly. The results are shown in Figure 4a where the horizontal axis represents the dataset sizes in the ascending order and the vertical axis is the corresponding computational time. The result of approximate CDOF was averaged over ten trials of data sampling. It is shown that approximate CDOF is faster than LOF and slower than EDOF. This reflects the complexities of $O(n)$, $O(nlogn)$, and $O(n^2)$ for EDOF, approximate CDOF, and LOF, respectively.

In order to validate the effectiveness of approximate CD, we used CD and approximate CD to find outliers in five synthetic datasets generated in the same



(a) Performances of EDOF, LOF, and approximate CDOF



(b) Effectiveness of approximate CDOF

Fig. 4. Performances of the method using approximate commute distance

way as the experiment noted above with smaller sizes due to the high computation of CDOF. The results were averaged over ten trials. The results in Figure 4b shows approximate CDOF (aCDOF) is much faster than CDOF but still preserves a high percentage (86.2% on average) of top outliers found by CDOF.

5.5 Impact of Parameter k

In this section, we investigate how the number of nearest neighbors affects CDOF. Denote k_{min} as the maximum number of nodes that a cluster is an outlying cluster and k_{max} as the minimum number of nodes that a cluster is a normal cluster. There are two situations. If we choose the number of nearest neighbors $k_2 < k_{min}$, nodes in an outlying cluster do not have neighbors outside the cluster. As a result, their outlier factors are small and we will miss them as outliers. On the other hand, if we choose $k_2 > k_{max}$, nodes in a normal cluster have neighbors outside the cluster. And it is possible that some nodes in the cluster will be falsely recognized as outliers. The value of k_{min} and k_{max} can be considered as the lower and upper bounds for the number of nearest neighbors. They can be different depending on the application domains. In the experiment in Section 5.1, we chose $k_2 = 15$, which is just greater than the sizes of all outlying clusters (i.e 12) and is less than the size of the smallest normal cluster (i.e 100). The same result can be obtained with $15 < k_2 < 100$ but it requires longer computational time. k_2 is also chosen as a threshold to distinguish between normal and outlying clusters.

Note that k_2 mentioned in this section is the number of nearest neighbors for estimating the outlier scores. For building mutual k_1-nearest neighbor graph, if k_1 is too small, the graph is very sparse and may not represent the dataset densities properly. In the experiment in Section 5.1, if $k_1 = 5$, the algorithm misclassifies some nodes in the smaller normal cluster as outliers. If k_1 is too large, the graph tend to connect together clusters whose sizes are less than k_1

and are close to each other. Then some outlying clusters may not be detected if they connect to each other and form a normal cluster. $k_1 = 10$ is found suitable for many synthetic and real datasets.

6 Conclusions

We have proposed a method for outlier detection using 'commute distance' as a metric to capture global, local outliers, and outlying clusters. The CD captures both distances between observations and their local neighborhood densities. We observed and proved a property of CD which is useful in capturing local neighborhood density. The experiments have shown the effectiveness of the proposed method in both synthetic and real datasets. Moreover, graph component sampling and eigenspace approximation used to approximate CD and the use of pruning rule can accelerate the algorithm significantly while still maintaining the accuracy of the detection. Furthermore preliminary experiments suggest that CDOF is less sensitive to perturbations in data than other measures.

Acknowledgement

The authors of this paper acknowledge the financial support of the Capital Markets CRC.

References

1. Database basketball, http://www.databasebasketball.com
2. Bay, S.D., Schwabacher, M.: Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In: KDD '03: Proc. of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 29–38. ACM, New York (2003)
3. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: Identifying density-based local outliers. In: Chen, W., Naughton, J.F., Bernstein, P.A. (eds.) Proc. of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas, USA, May 16-18, pp. 93–104. ACM, New York (2000)
4. Chandola, V., Banerjee, A., Kumar, V.: Outlier detection: A survey. Tech. Rep. TR 07-017, Department of Computer Science and Engineering, University of Minnesota, Twin Cities (2007)
5. Chung, F.: Spectral Graph Theory. In: Conference Board of the Mathematical Sciences, Washington. CBMS Regional Conference Series, vol. 92 (1997)
6. Fouss, F., Renders, J.M.: Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Transaction on Knowledge and Data Engineering 19(3), 355–369 (2007)
7. Hawkins, D.: Identification of Outliers. Chapman and Hall, London (1980)
8. Khoa, N.L.D., Chawla, S.: Unifying global and local outlier detection using commute time distance. Tech. Rep. 638, School of IT, University of Sydney (2009)
9. Klein, D.J., Randic, M.: Resistance distance. Journal of Mathematical Chemistry 12, 81–95 (1993), http://dx.doi.org/10.1007/BF01164627

10. Knorr, E.M., Ng, R.T.: Algorithms for mining distance-based outliers in large datasets. In: The 24rd International Conference on Very Large Data Bases, pp. 392–403 (1998)
11. Lovász, L.: Random walks on graphs: a survey. Combinatorics, Paul Erdös is Eighty 2, 1–46 (1993)
12. Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing 17(4), 395–416 (2007)
13. Moonesinghe, H.D.K., Tan, P.N.: Outrank: a graph-based outlier detection framework using random walk. International Journal on Artificial Intelligence Tools 17(1), 19–36 (2008)
14. Rousseeuw, P.J., Leroy, A.M.: Robust Regression and Outlier Detection. John Wiley and Sons, Chichester (2003)
15. Saerens, M., Fouss, F., Yen, L., Dupont, P.: The principal components analysis of a graph, and its relationships to spectral clustering. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 371–383. Springer, Heidelberg (2004)
16. Sun, P.: Outlier Detection In High Dimensional, Spatial And Sequential Data Sets. Ph.D. thesis, The University of Sydney (2006)

EigenSpokes: Surprising Patterns and Scalable Community Chipping in Large Graphs*

B. Aditya Prakash[1], Ashwin Sridharan[2], Mukund Seshadri[2], Sridhar Machiraju[3], and Christos Faloutsos[1]

[1] Computer Science Department, Carnegie Mellon University
`{badityap,christos}@cs.cmu.edu`
[2] Sprint Applied Research Labs, California
`{ashwin.sridharan,mukund.seshadri}@sprint.com`
[3] Google Inc., California
`machiraju@acm.org`

Abstract. We report a surprising, persistent pattern in large sparse social graphs, which we term *EigenSpokes*. We focus on large Mobile Call graphs, spanning about *186K* nodes and *millions* of calls, and find that the singular vectors of these graphs exhibit a striking *EigenSpokes* pattern wherein, when plotted against each other, they have clear, separate lines that often neatly align along specific axes (hence the term "spokes"). Furthermore, analysis of several other real-world datasets *e.g.,* Patent Citations, Internet, *etc.* reveals similar phenomena indicating this to be a more fundamental attribute of large sparse graphs that is related to their community structure.

This is the first contribution of this paper. Additional ones include (a) study of the conditions that lead to such *EigenSpokes*, and (b) a fast algorithm for spotting and extracting tightly-knit communities, called *SpokEn*, that exploits our findings about the *EigenSpokes* pattern.

1 Introduction

Given a large phone-call network, how can we find communities of users? While the behavior of users in landline networks has been examined before [4], we study here the phone call network of mobile users in cellular networks. The analysis of mobile phone graphs is interesting for multiple reasons as mobile phones are ubiquitous and are a key conduit for Internet access too. Several recent studies have used mobile call graph data to examine and characterize the social interactions of cell phone users, with a focus on understanding the structural properties of the graph [13,11,21], the evolution of social groups and the spread of new products and services [14].

* This material is based upon work supported by the National Science Foundation under Grants No. CNS-0721736 and CNS-0721889 and a Sprint gift. Research partly done during a summer internship by the first author at Sprint Labs. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

Our objective in this paper is to identify if and to what extent do well-defined social groups of callers exist in such networks. We emphasize that understanding the *entire* graph structure is *not* our goal. Indeed, in large social networks, not *every* node can be expected to belong to a community. Hence, extraction of community-like structures, which can be independently analyzed, is the focus of this paper rather than graph-partitioning. This approach based on chipping off communities is also supported by recent studies [10] that have shown the presence of small communities loosely connected with the remaining "core" of the graph. Furthermore, when we applied well-known graph clustering techniques on our datasets, none of them provided much insight into chipping off interesting community structures for further analysis, since these techniques are geared towards partitioning the *entire* graph. What was surprising, though, was our discovery of the '*spokes*' (or *EigenSpokes*) phenomenon (see Figure 1(b)): the singular vectors of the Mobile Call graph, when plotted against each other, often have clear separate lines, typically aligned with axes. We term such plots EigenEigen (or *EE*) plots.
We concentrate on three key questions in this paper:

1. **Cause:** What causes these *spokes*?
2. **Ubiquity:** Do they occur across varied datasets to be worth studying?
3. **Community Extraction:** How can we exploit them, to chip off meaningful communities from large graphs?

We answer these three questions on graphs of Table 1. Our primary dataset in this paper is an anonymized social graph based on mobile calls made from/to callers located within a geographically contiguous urban area. In this social graph, callers are represented as nodes, and edges represent calls between nodes. This Mobile Call graph captures activity over the duration of a month (millions of successfully completed calls) and consists of about $186,000$ nodes and $464,000$ edges. We also investigate similar Mobile Phone graph datasets obtained from other geographic areas, to support our findings. Since these graphs are disconnected, we focus on the largest connected component. Our graphs exhibit characteristics such as degree distributions and generative processes similar to those of other mobile call graphs [11,21].

Table 1. Graph datasets used in this paper

Name	Description	Nodes	Edges
Mobile Call graph	Calls between callers/callees	186,000	464,000
Patent Citations	Citations between patents	3,774,768	14,970,767
Internet routers	Network links between routers	124,651	207,214
Dictionary words	Words are connected if they differ by a single letter	52,652	178,076

In addition to the above, we also investigate several other datasets (Table 1) in the public domain[1] which allow us to determine the generality of our observations

[1] http://www.cise.ufl.edu/research/sparse/matrices/

and the underlying phenomenon. Also, they have meta-information that helps us demonstrate that our algorithm chips-off meaningful communities.

The following sections discuss the related work, problems with traditional methods, explain the *EigenSpokes* pattern, develop the *SpokEn* algorithm and finally present many surprising communities found in the datasets.

2 Related Work

Graph partitioning is a popular approach for studying community structure in graphs. Popular methods include Spectral clustering (see [24] for a survey), a "cut-based" method for understanding graph structures, which has been successful in machine-learning and image segmentation. Similar approaches (e.g. [19,16]) use the eigenvectors of the adjacency matrix. Lastly, spectral inspired methods have been used to learn model parameters for well-separated Gaussian mixtures [22]. Alternative cut-based multilevel approaches like Metis [8] and Graclus [5,20] coarsen the graph by coalescing nodes and then apply refinement steps to recover partitions. We address both the Spectral and multi-level partitioning techniques in greater detail in § 3.

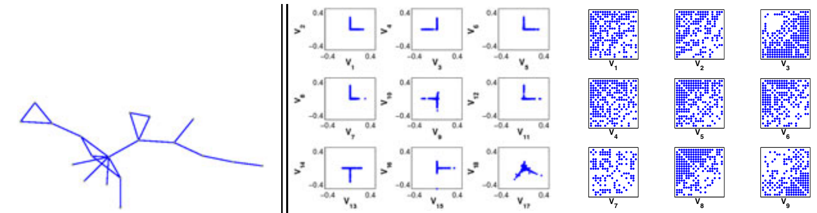
Cross-Association [2] partitions the graph so as to maximize information compression, but is limited to bi-partite structures. Co-clustering [6] trys to maximize mutual information, but like k-means, requires *a priori* information on the number of clusters. More generally in terms of clique extraction, [15] trys to extract quasi-cliques from graphs. Our focus however is on *chipping* out general community structures.

Modularity based approaches compare a graph's community structure against a random graph. Studies have proposed using modularity based Laplacian-like matrices [12,25] or greedy heuristics [3] for graph clustering. However all these techniques also partition the *entire* graph rather than extract relevant communities, which is our objective.

In terms of social network analysis, [18] extracts communities from an Instant Messenger Network by applying Co-clustering. [7] proposes flow-based techniques to identify Web-based communities. However, it identifies communities for a set of *known* nodes, while our objective is to extract all nodes that constitute communities. Also, although the focus of [26] is on randomness measures, they observed quasi-orthogonal spectral lines in context of small cavemen-like graphs. To the best of our knowledge the *EigenSpokes* pattern has not been observed in any *real*, *large* social networks.

3 Why Not Traditional Methods?

We analyzed the mobile call graph using well-known spectral clustering [24] and multi-level graph partitioning [8,5] techniques. Our goal here is to explore if these methods can help us extract communities of nodes for further analysis. Although remarkably successful in other settings like image segmentation etc., we find, as shown below, that these methods do not yield good communities in our graph.



(a) L_{rw}: **Note the long chains** (b) **EE-plot** (c) **Spy Plots of subgraph of Top 20 Nodes**

Fig. 1. (a) Typical Partition using L_{rw} (b),(c) ***EigenSpokes*** in Region 1 & Time 1

Spectral Clustering: L_{rw}, L_{sym}, ... Many "Laplacian" matrices can be defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ (see [24]). We applied the L_{rw} method [24] on our primary data set to obtain k-way partitions from $k = 2$ to $k = 100$. As in recent studies ([9], [10]), we found that the application of the technique yields :a) skewed partitions consisting of very small clusters and a large 'core' and b) the clusters lack internal coherence. The partitions we got from L_{rw} while lowering the N-cut value had no or little internal coherence with long chains and most nodes connected to 1 or 2 other nodes (see Figure 1(a)). We experimented with several other Laplacians including L_{sym} [24] and L_Q (based on modularity) [25] but obtained similar results of limited utility.

Graph Partitioning Methods Prevailing multilevel algorithms for graph partitioning like Metis [8] and their improvements like Graclus [5] and MCR-MCL [20] are based on repeated coarsening and refinements of nodes with emphasis on balanced cuts. To explore how well multilevel algorithms perform, we ran Graclus on the Mobile Call graph to obtain k partitions. We invoked the algorithm with various values for k from $k = 2$ to $k = 10,000$. While Graclus yields more balanced clusters than Spectral partitioning, we observed that, as before, the clusters lack internal coherence. This can be attributed, again, to the following two causes : a) these algorithms also utilize a cut-based metric and b) their objective is to partition the *entire* graph; as shown by our results as well as [10], this is not feasible when applied to social graphs that comprise of a large set of random nodes and small communities.

4 EigenSpokes

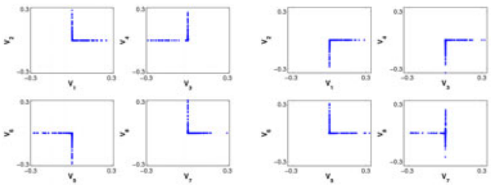
As demonstrated in the previous section (and prior work), Laplacians in certain large graphs yield communities that have a low cut but possess little internal coherence. Hence, we investigate using the adjacency matrix itself. This leads to several interesting observations, and motivates our approach for community identification.

4.1 A Surprise: Spokes

Recall that the *Singular Value Decomposition* (SVD) of an $m \times n$ matrix W is a factorization defined as: $W = U\Sigma V^T$, where U and V are $m \times m$ and

$n \times n$ size matrices respectively, and Σ is an $m \times n$ diagonal matrix comprised of the singular values. Taking the top K values of Σ yields the best rank-K approximation (w.r.t. the Frobenius norm) to the original matrix [23].

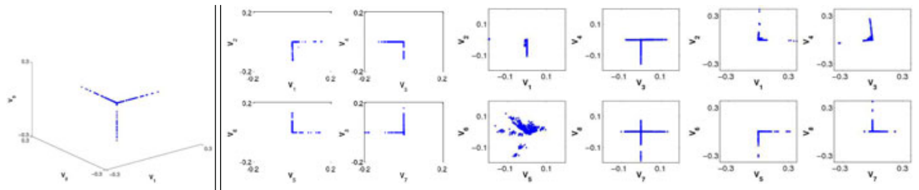
We define the *EE-plot* as the scatter plot of vector U_i and U_j, for any i and j, i.e., they plot one point (U_{in}, U_{jn}) for each node n in the graph. Surprisingly, we find that the *EE-plots* for our Mobile Call graph show clear separate straight lines that are often aligned with axes[2]! We call this the *EigenSpokes* pattern. This is demonstrated in Figure 1(b), where we plot the first $K = 18$ singular vectors pairwise. Even more striking is that, as shown in Figure 2, these spokes occur in many Mobile Call graphs collected at various points of time (separated by several months) and at various geographic regions.



(a) **Region 2, Time 2** (b) **Region 3, Time 3**

Fig. 2. *EE-plots* for Mobile Call graphs for different geographic regions and time periods: note the persistence of *EigenSpokes*

Some intuition: We delve further into the *EigenSpokes* pattern by identifying the nodes that lie on the extremities of the "spokes"; more precisely, for each of the first 9 singular vectors, we identify the 20 nodes that had the highest magnitude projection along that vector. We then plot the induced sub-graph of these nodes (see Figure 1(c)). Clearly, almost all of the induced sub-graphs contain near-cliques. *These observations hint toward a strong connection between EigenSpokes and communities, and raise the following questions : are these spokes representative of fundamental community structures; do they occur elsewhere? What is their origin and how exactly can they be used for chipping off communities?*



(a) (b) **Patents** (c) **Dictionary** (d) **Internet**

Fig. 3. (a) *EEE*-plot for Mobile Call graph and *EigenSpokes* in other Real-World Data Sets (b),(c),(d)

We answer these questions next: in § 4.2, we provide a rigorous basis for the link between *EigenSpokes* and communities; in § 4.3, we show that *EigenSpokes* can be observed in several real-world graphs; and in § 4.4, we demonstrate the various conditions that lead to the presence (and absence) of the *EigenSpokes* phenomenon.

[2] Axis-alignment is distinct from orthogonality of the vectors.

4.2 Justification and Proofs

Our focus in this paper is on undirected graphs, which implies that the adjacency matrix A is a square, symmetric matrix. For such a matrix, it is well known that the singular values are absolute values of the non-zero eigenvalues: $\sigma_i = |\lambda_i|$ and the singular vectors coincide with the non-null eigenvectors. Due to the equivalence between singular vectors and eigenvectors for the (symmetric) graphs considered in this paper, we abuse language and often refer to singular vectors as being the spectra of A.

Given the presence of *EigenSpokes*, is it reasonable to expect the nodes lying along the extremes to have similar connectivity patterns?

***EigenSpokes*, Connectivity and Communities:** The presence of spokes in *EE-plots* (axis-aligned or not) implies that nodes close to each other on a line have similar scores along *two* eigenvectors ('score' of node n along vector U_i is U_{in}). In fact, plots of the first 3 eigenvectors, or *EEE-plots* (see Figure 3) show lines too. This strongly suggests similar scores for the nodes in *many* vectors. Specifically, consider two nodes i and j whose connectivity information is represented by their rows A_i and A_j in A. If the k^{th} eigenvector is denoted by U_k, then $A_i U_k$ and $A_j U_k$ are the i^{th} and j^{th} components of U_k. These will be equal if $A_i = A_j$. Hence, the two nodes will have the same components along the eigenvectors. In general, we expect that *nodes with similar connectivity will have similar scores along the vectors of U.*

Is the converse also true? We can prove the following lemma to this end (note that $\langle x, y\rangle = x^T y$ denotes the dot-product of two column vectors x and y):

Lemma 1. *For any real, symmetric adjacency matrix A, if for any i and j, $\forall k,\ |\langle (A_i - A_j)^T, U_k\rangle| \leq \epsilon$, then $\forall k,\ |A_{ik} - A_{jk}| \leq (\epsilon\sqrt{N})$ as well.*

Proof. As A is real symmetric, by the Spectral Theorem, it is orthogonally diagonalizable. Hence, it is non-defective and has a full basis of eigenvectors. Consider any vector $C = \sum_k c_k U_k$ written using the basis consisting of the eigenvectors. Then,

$$\langle (A_i - A_j)^T, C\rangle \;=\; \sum_k c_k \langle (A_i - A_j)^T, U_k\rangle$$

$$\leq\; \sqrt{\sum_k c_k^2}\sqrt{\sum_k \langle (A_i - A_j)^T, U_k\rangle^2} \;\leq\; \sqrt{\sum_k c_k^2}\sqrt{N}\ \epsilon$$

where we use the Cauchy-Schwartz inequality in step 2 and given bound in step 3. Use the above inequality for $C = e_k$ (indicator vector which is zero everywhere except at index k where it is 1), for every k. Also, note that orthogonal transformations preserve the norm of a vector - hence, $\sqrt{\sum_k c_k^2} = \| C \|$, which would be equal to 1 for our choice of C's. Therefore, we get: $\forall k, |A_{ik} - A_{jk}| \leq (\epsilon\sqrt{N})$. □

Note that the above proof holds for any orthogonal basis set of vectors U; but our basis set U is also a carefully chosen set: it is the set of singular vectors. Hence, we expect the bound to be tighter in practice.

In view of the above, we expect that nodes lying close to each other on a spoke will have similar neighbor sets. But what is the link between similar neighbor sets and communities in the graph? Consider the following two (sufficient) conditions that result in similar neighbor sets:

1. Cliques (or near-cliques) result in exactly the same [3] (or similar) adjacency rows for nodes in the cliques.
2. Perfect (or near-perfect) Bipartite-cores also result in the same (or similar) adjacency rows for nodes in the two cores.

Consequently, we expect to see communities in the form of (near-)cliques or (near-)bi-partite cores among the nodes in the spokes.

Axis-alignment (or not): Recall another striking feature of the *EE-plots*: the presence of largely *axis-aligned* spokes. From the preceding discussion, given the presence of *EigenSpokes*, we should traverse the extreme points on each spoke to extract communities. This reduces to searching for 'high-scoring' nodes in each singular vector *separately*, since axis-alignment implies that the extreme points have high values only in one of the vectors. In fact, prior work [1] has already shown that nodes with scores at the extreme ends of the principal eigenvector of Erdos-Renyi graphs do belong to the same clique. However, some spokes are not axis-aligned (as in the last *EE*-plot of Figure 1(b)). This implies that some nodes have significant scores along multiple singular vectors. In the specific case of *EE-plots* the linear nature of spokes means that the scores of the nodes are *linearly correlated*. Hence, exploring dominant nodes along one singular vector should be sufficient to extract nodes of a community.

4.3 Ubiquity of Spokes

Apart from Mobile Call graphs, we have observed the *EigenSpokes* pattern with singular vectors of several other real world graphs. We show the *EE-plots* for Patents, Dictionary and Internet router connectivity in Figure 3(b), 3(c) and 3(d) ([17] contains more detailed plots). In all three cases, we see that most pairwise combinations align in a spoke pattern, with some exceptions in the Dictionary graph. We also observe that some of these spoke patterns are not axis-aligned; as discussed earlier though, the linear correlation between the scores of the nodes is preserved. Thus the *EigenSpokes* pattern is persistent across a wide array of diverse datasets.

4.4 Recreating Spokes

So far, we have provided some insight into why spokes arise. We now demonstrate exactly which features of graphs and community structure result in spokes using both synthetic and real graphs. Synthetic graphs, in particular, allow us to experiment with various parameters and characteristics, and observe their effect on their *EE*-plots. We show that the key factors responsible for these patterns are a **large** number of **well-knit** communities embedded in very **sparse** graphs.

[3] Not considering self-edges.

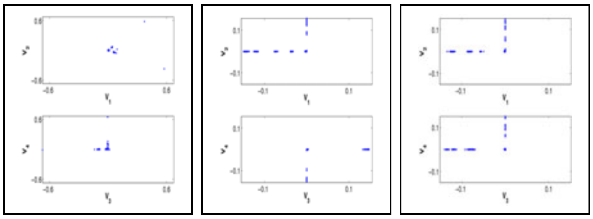

(a) **No Communities** (b) **40 Communities** (c) **400 Communities**

Fig. 4. Genesis of *EigenSpokes*: synthetic heavy-tailed random graphs with community structure

We started with a synthetic random heavy-tailed graph with the same number of nodes and degree distribution as our Mobile Call graph but with no community structure. The EE-plots don't exhibit any spokes pattern (Figure 4(a)) but, when we synthetically introduce 40 communities (near-cliques of sizes $31 - 50$, with a probability 0.8 of an intra-community edge) into the above random graph, in Figure 4(b), we observe the emergence of the spokes pattern. When we increase the number of communities to 400, in Figure 4(c), the spoke pattern becomes more clear, and resembles Figure 2. Further, we verified that the nodes at the extremities do indeed form the artificially embedded communities. We also found that the nature of the communities, including the level of internal connectivity, does not affect the emergence of the spokes pattern as long as such connectivity is significant. Thus we infer that one of the important causes for a spokes pattern is the presence of a **large number of tightly knit communities** in the graph.

Due to lack of space we omit details about the effect of sparsity and degree density; they are demonstrated in greater detail in [17].

5 *SpokEn*: Exploiting EigenSpokes

Based on the insights from previous sections, we now develop our community identification approach *SpokEn*, that exploits *EigenSpokes*. While developing *SpokEn*, we also use experiments with synthetically generated graphs to help us choose from the various algorithmic choices. Our approach differs from prior work on graph partitioning as for certain classes of graphs, we observe a specific structural property and exploit it. While this may not apply to all graphs, our approach is highly effective for the large sparse graphs we consider, as shown by our results later in the paper.

5.1 Designing *SpokEn*

Our proposed approach is based on the key property of *EigenSpokes* highlighted in § 4.2: the existence of *EigenSpokes* indicates the presence of well-knit communities whose nodes have a significant component in *that* singular vector. Thus an appropriate traversal of each singular vector in *isolation* can extract these communities. A good traversal should select only the nodes which belong to a coherent community. We now discuss where to start the traversal, how to *grow* the community and finally, when to stop.

Initialization: We choose the node with the score of maximum magnitude as the *seed* for the community. We multiply the given singular vector U_i by -1 if necessary to ensure that the score with the largest magnitude is positive.

Discovery: A simple algorithm for discovery is one that picks nodes in decreasing order of their scores. Such an algorithm can pick a node that is disconnected from all the nodes chosen previously. Hence, we propose the following: let C denote the set of all nodes that have been discovered so far; the next node that we select is the node with the largest score that is connected to some node in C. Formally, we augment C with a node n^* that satisfies $n^* = \arg\max_{n \in N_C} U_i(n)$, where N_C is the neighborhood of C[4]. This algorithm is intuitive and keeps C always connected.

Termination and Trimming: For termination, we need to use a metric that quantifies the quality of the community extracted so far. We propose to use a novel hybrid approach based on conductance [24] for cut and modularity (actually *relative* modularity) for coherence. The process discovers and adds nodes to the set C as long as the relative modularity increases and terminates once it reduces indicating reduction in community structure. We finally use a conductance based method to trim out the remaining false positives.

5.2 Discussion

Relative Modularity: In large graphs such as ours, underlying communities are typically small ($10 \approx 100$ in a million node-graph) [10]. The equation for modularity[5] indicates that when extracting a *single* small community from a large graph, the modularity metric computed on such a highly unbalanced partition would be dominated by the larger partition and *not* the discovered community, thus rendering it useless. This was also empirically observed in extensive evaluations over our datasets.

To resolve this problem, we once again utilize the concept of scores. When traversing a singular vector (say U_i), as a pre-processing step, we construct a *new* sub-graph $G_\epsilon = (\mathcal{V}_\epsilon, E_\epsilon)$ from G wherein we discard all nodes n with values $U_i(n)$ *below* a certain threshold ϵ. The modularity computation is then conducted w.r.t. G_ϵ (hence *relative* modularity). This is justified since in the first place, we do not expect the discarded nodes to belong to the community under consideration. The removal of such nodes induces a more balanced partition and makes the modularity values more meaningful. We set $\epsilon = 10^{-4}$ in our experiments as several tests showed the results were insensitive around it.

Trimming using Conductance: As shown later in § 5.3, conductance as a termination criterion results in *premature* termination of the discovery process,

[4] N_C is the set of nodes not in C and are connected to at least one node in C.

[5] The modularity of a graph partition $\mathcal{C} = \{C_1, C_2, \ldots\}$ is $Q(C) = \frac{1}{2m} \sum_{C \in \mathcal{C}} \sum_{i,j \in C \cap \mathcal{V}} \left[A_{ij} - \frac{k_i k_j}{2m} \right]$, where k_i is the degree of a node and A_{ij} an element of the adjacency matrix.

causing several false negatives while relative modularity as a termination metric often results in *overshooting* and hence false positives. These two observations indicate that modularity and conductance are *complementary* in the role of a termination metric which is why we adopt a hybrid approach. Hence as mentioned before, after a community is extracted using relative modularity (to discover all relevant nodes at the cost of false positives), a standard spectral technique (L_{rw}) is used to trim-out false positives by further bisection to determine a better cut. We give the pseudo-code of *SpokEn* in Algorithm 1.

Algorithm 1. *SpokEn*

```
Require: Symmetric binary adjacency matrix $A$
 1: $U$ = get first several eigenvectors of $A$
 2: Stop  if $U$ has no EigenSpokes pattern
 3: for all eigenvectors $v = U_k$ do
 4:    Construct $G_\epsilon = (\mathcal{V}_\epsilon, E_\epsilon)$ such that $\mathcal{V}_\epsilon = \{i : v(i) > \epsilon\}$
 5:    Initialize outputSet using seed   //see Initialization
 6:    //see relative modularity in Termination
 7:    while modularity(outputSet, $G_\epsilon$) increases do
 8:       Expand outputSet   //see Discovery
 9:    end while
10:    $C_k$ = trim outputSet using conductance   //see Triming using Conductance
11: end for
12: return $\{C_k\}$
```

5.3 Empirical Results

To evaluate the performance of our discovery process and termination criterion, we applied *SpokEn* on various synthetically generated graphs with known ground truths in each case (like the ones described in § 4.4 including an ER graph embedded with *bi-partite cores*). While the detailed results are provided in [17], we found that conductance as a termination criterion undershot and hence detected fewer communities (about 60% with almost no false positives). On the other hand, modularity discovered about 80% communities but with 4% false positives. Their combination, *SpokEn*, was able to identify 76-90% of the embedded communities with almost no false positives.

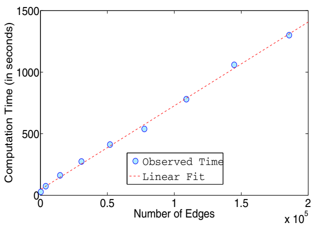
Speed : The computation time is mainly dominated by the eigenvector calculation which is linear in edges. We ran *SpokEn* on graphs of various sizes on a Dell Server with an Intel Xeon 3 GHz processor and 4 GB of RAM. In each case, we computed 100 singular vectors and mined communities from them. Figure 5(a) plots the computation time required by *SpokEn* to extract communities as a function of the number of edges of the graph. As expected, it clearly shows the processing time is *linear* in the number of graph edges, which is a key indication of scalability.

6 Successes with Real-World Graphs

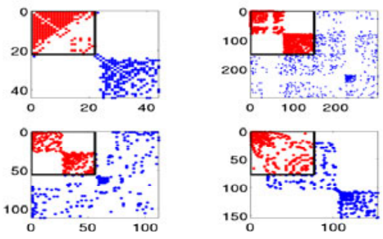
We applied *SpokEn* to our real-world datasets and found that it extracts several interesting communities that reveal useful and relevant information about the connectivity patterns.

We illustrate four typical communities extracted by *SpokEn* from our Mobile Call graph. Figure 5(b) presents the spy plots which clearly show that the communities are well-connected (the communities are red nodes on the top delineated by black boxes). To show the success of our terminating criterion, we plot additional nodes that would have been explored by the discovery process without termination. Notice that *SpokEn* does indeed typically stop at points where a community appears to have ended. In the bottom left case, however, it overshoots and combines what appear to be two near-cliques into the same community. We observe similar results for the other singular vectors as well: *SpokEn* extracts communities of nodes with good internal coherence though it sometimes clubs together two such communities into one.

The prevalence of such close-knit communities of more than 10 nodes in a Mobile Call graph was quite unexpected to us. Temporal analysis of the usage of at least a few communities leads us to believe that these communities arise from users of the same organization.



(a) **Scalability of *SpokEn***



(b) **Communities by *SpokEn***

Fig. 5. (a) *SpokEn*: Computation Time is linear in #(edges) **(b)** Four communities (delimited by black box and red dots) extracted by *SpokEn* from Mobile Call graph (spy plots). To illustrate the success of the termination criterion, we plot additional nodes in the order of discovery. Notice the near-clique (= near block diagonal) behavior.

Next, we analyze a few typical communities from other data sets. Figure 6(a) plots the connectivity between nodes of a community extracted from the Patent citation dataset. Notice the striking bipartite nature of the community. Upon further investigation, we find that the bipartite nature arises because of patents filed by the *same* organization on related topics reuse the bibliography entries. In the example shown here, one-half of the bi-partite graph comprises of about 25 patents that were filed in a period of 1998 – 1999 by (the same) authors from Kimberly-Clark in the area of photosensitive pigments for color printers. The bi-partite nature of the graph arises because all these 25 patents cited the *same* set of past references. This also illustrates a crucial aspect of *SpokEn*: it extracts communities of nodes with *similar* connectivity. This may or may not imply *mutual* connectivity.

Figure 6(b) shows a typical community extracted from the Dictionary graph. Recall that this dataset connects two words if they differ by exactly one letter. The clique shown in Figure 6(b) arises from three-letter words that all end with 'on'. We found many similar cases including words that end with 'an', 'll', etc. Finally, Figure 6(c) shows a community extracted from the router graph. The community highlights the tiering relationship typical in the Internet. The community consists of 4 UUNET back-bone routers (first and third row) from the Tier-1 layer that serve as gateways for a large community of Tier-2 Verizon Business and other small business (the second row) and are also connected to other Tier-1 routers (Sprint, AT&T *etc.*, last row).

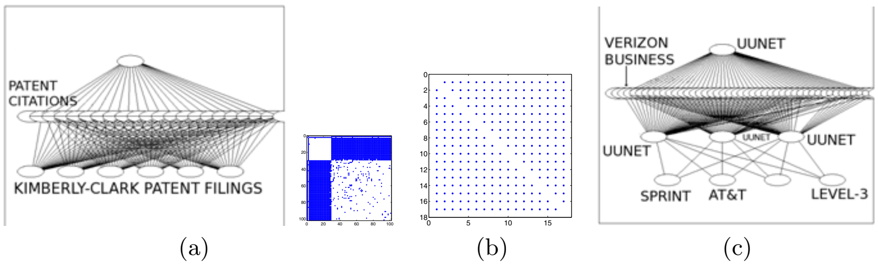


Fig. 6. Structures extracted by *SpokEn* from real-world graphs. **(a)** A typical bi-partite community extracted from the Patent graph. It arises due to "cut-and-paste" bibliography generation. On the left, we plot a portion of the community for visual understanding, and the entire spy plot on the right. **(b)** A typical near-clique subgraph of words from the Dictionary graph. The words all differ by exactly one letter from each of the others. **(c)** Internet router connectivity from one provider to customers causing a bipartite community.

7 Conclusions

In answer to the questions we posed earlier in § 1, we find that:

1. **Cause:** *Spokes* can be strongly associated with the presence of well-defined communities like cliques and bi-partite cores in sparse graphs.
2. **Ubiquity:** Apart from Mobile Call graphs, they occur in a variety of data-sets such as Patent citations, Dictionary and Internet.
3. **Community Extraction:** The *spokes* pattern allows us to construct an efficient and scalable algorithm "*SpokEn*" that helps us chip off communities thereby revealing several interesting structures in Mobile Call graphs as well as the other datasets.

References

1. Alon, N., Krivelevich, M., Sudakov, B.: Finding a large hidden clique in a random graph. In: Proc. of the Ninth Annual ACM-SIAM SODA (1998)
2. Chakrabarti, D., Papadimitrou, S., Modha, D., Faloutsos, C.: Fully automatic cross-associations. In: Proc. of the tenth ACM SIGKDD (2004)
3. Clauset, A., Newman, M.E.J., Moore, C.: Finding Community Structure in Very Large Networks. Physical Review (2004)
4. Cortes, C., Pregibon, D., Volinsky, C.: Communities of interest. In: Hoffmann, F., Adams, N., Fisher, D., Guimarães, G., Hand, D.J. (eds.) IDA 2001. LNCS, vol. 2189, pp. 105–114. Springer, Heidelberg (2001)
5. Dhillon, I., Guan, Y., Kullis, B.: Weighted Graph Cuts without EigenVectors: A Multilevel Approach. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1944–1957 (November 2007)
6. Dhillon, I., Mallela, S., Modha, D.: Information-theoretic co-clustering. In: Proc. of the ninth ACM SIGKDD (2003)
7. Flake, G., Lawrence, S., Giles, L.: Efficient identification of web communities. In: Proc. of Sixth ACM KDD (2003)
8. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Jnl. on Sci. Comp. 20(1), 359–392 (1999)
9. Lang, K.: Fixing two weaknesses of the spectral method. In: Proc. of NIPS (2006)
10. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Statistical properties of community structure in large social and information networks. In: WWW (2008)
11. Nanavati, A.A., Gurumurthy, S., Das, G., Chakraborty, D., Dasgupta, K., Mukherjea, S., Joshi, A.: On the structural properties of massive telecom call graphs: findings and implications. In: Proc. of 15th ACM CIKM, pp. 435–444 (2006)
12. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E 74(036104) (2006)
13. Onnela, J.-P., Saramaäki, J., Hyvöven, J., Szabó, G., de Menezes, M.A., Kaski, K., Barabási, A.-L.: Structure and Tie Strengths in Mobile Communication Networks. New Journal of Physics 9 (2007)
14. Palla, G., Barabasi, A.-L., Vicsek, T.: Quantifying social group evolution. Nature 446(664) (2007)
15. Pei, J., Jiang, D., Zhang, A.: On mining cross-graph quasi-cliques. In: Proc. of the eleventh ACM SIGKDD (2005)
16. Perona, P., Freeman, W.T.: A factorization approach to grouping. In: Burkhardt, H.-J., Neumann, B. (eds.) ECCV 1998. LNCS, vol. 1406, pp. 655–670. Springer, Heidelberg (1998)
17. Prakash, B.A., Sridharan, A., Seshadri, M., Machiraju, S., Faloutsos, C.: Patterns and community extraction in large graphs. TR CMU-CS-09-176 (2009)
18. Resig, J., Dawara, S., Homan, C., Teredesai, A.: Extracting social networks from instant messaging populations. In: Proc. of ACM SIGKDD (2004)
19. Sarkar, S., Boyer, K.L.: Quantitative measures of changed based on feature organization: Eigenvalues and eigenvectors. Computer Vision and Image Understanding 71(1), 110–136 (1998)
20. Satuluri, V., Parthasarathy, S.: Scalable graph clustering using stochastic flows: applications to community discovery. In: Proc. of ACM SIGKDD (2009)
21. Seshadri, M., Machiraju, S., Sridharan, A., Bolot, J., Faloutsos, C., Leskovec, J.: Mobile call graphs: Beyond power-law and lognormal distributions. In: KDD (2008)

22. Shi, T., Belkin, M., Yu, B.: Data Spectroscopy: Learning Mixture Models using Eigenspaces of Convolution Operators. In: Proc. of ICML (2008)
23. Strang, G.: Introduction to Linear Algebra. Wellesley-Cambridge Press (2003)
24. von Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing 17(4), 395–416 (2007)
25. White, S., Smyth, P.: A spectral clustering approach to finding communities in graphs. In: Proc. of SDM (2005)
26. Ying, X., Wu, X.: On randomness measures for social networks. In: SDM (2009)

BASSET: Scalable Gateway Finder in Large Graphs

Hanghang Tong[1], Spiros Papadimitriou[2], Christos Faloutsos[1], Philip S. Yu[3], and Tina Eliassi-Rad[4]

[1] Carnegie Mellon University
{htong,christos}@cs.cmu.edu
[2] IBM T.J. Watson
spapadim@us.ibm.com
[3] University of Illinois at Chicago
psyu@cs.uic.edu
[4] Lawrence Livermore National Laboratory
eliassi@llnl.gov

Abstract. Given a social network, who is the best person to introduce you to, say, Chris Ferguson, the poker champion? Or, given a network of people and skills, who is the best person to help you learn about, say, wavelets? The goal is to find a small group of 'gateways': persons who is close enough to us, as well as close enough to the target (person, or skill) or, in other words, are crucial in connecting us to the target.

The main contributions are the following: (a) we show how to formulate this problem precisely; (b) we show that it is sub-modular and thus it can be solved near-optimally; (c) we give fast, scalable algorithms to find such gateways. Experiments on real data sets validate the effectiveness and efficiency of the proposed methods, achieving up to *6,000,000x* speedup.

Keywords: Gateway, Sub-modularity, Scalability, Graph-Mining.

1 Introduction

What is the best gateway between a source node and a target node, in a network? This is a core problem that appears under several guises, with numerous generalizations. Motivating applications include the following:

1. In a corporate social network, which are the key people that bring or hold different groups together? Or, if seeking to establish a cross-division project, who are the best people to lead such an effort?
2. In an immunization setting, given a set of nodes that are infected, and a set of nodes we want to defend, which are the best few 'gateways' we should immunize?
3. Similarly, in a network setting, which are the gateway nodes we should best defend against an attack, to maximize connectivity from source to target.

The problem has several, natural generalizations: (a) we may be interested in the top k best gateways (in case our first few choices are unavailable); (b) we may have more than one source nodes, and more than one target nodes, as in the immunization setting above; (c) we may have a bi-partite graph with relationships (edges) between different node types, as in the last example above. Our main contributions in this paper are:

– A novel 'gateway-ness' score for a given source and target, that agrees with human intuition. Its generalization to the case where we have a group of nodes as the source and the target;
– Two algorithms to find a set of nodes with the highest 'gateway-ness' score, which (1) are fast and scalable; and (2) lead to *near-optimal* results;
– Extensive experimental results on real data sets, showing the effectiveness and efficiency of the proposed methods.

The rest of the paper is organized as follows: We give the problem definitions in Section 2; present 'gateway-ness' scores in Section 3; and deal with the computational issues in Section 4. We evaluate the proposed methods in Section 5. Finally, we review the related work in Section 6 and conclude in Section 7.

Table 1. Symbols

Symbol	Definition and Description
$\mathbf{A}, \mathbf{B}, \ldots$	matrices (bold upper case)
$\mathbf{A}(i,j)$	the element at the i^{th} row and j^{th} column of matrix $\mathbf{A}$
$\mathbf{A}(i,:)$	the i^{th} row of matrix $\mathbf{A}$
$\mathbf{A}(:,j)$	the j^{th} column of matrix $\mathbf{A}$
$\mathbf{A}'$	transpose of matrix $\mathbf{A}$
$\mathbf{a}, \mathbf{b}, \ldots$	column vectors
$\boldsymbol{p}, \boldsymbol{q}, \ldots$	ordered sequences
$\mathcal{S}, \mathcal{T}, \ldots$	sets (calligraphic)
n	number of nodes in the graph
m	number of edges in the graph
$\mathrm{g}(s,t,\mathcal{I})$	the 'Gateway-ness' score for the subset of nodes $\mathcal{I}$ wrt s and t
$\mathrm{g}(\mathcal{S},\mathcal{T},\mathcal{I})$	the 'Gateway-ness' score for the subset of nodes $\mathcal{I}$ wrt $\mathcal{S}$ and $\mathcal{T}$
$\mathbf{r}(s,t)$	the proximity score from s to t
$\mathbf{r}_{\mathcal{I}}(s,t)$	the proximity score from s to t by setting the subset of nodes indexed by $\mathcal{I}$ as sinks

2 Problem Definitions

Table 1 lists the main symbols we use throughout the paper. In this paper, we focus on directed weighted graphs. We represent the graph by its normalized adjacency matrix ($\mathbf{A}$). Following standard notation, we use capital bold letters for matrices (e.g., $\mathbf{A}$), lower-case bold letters for vectors (e.g., $\mathbf{a}$), and calligraphic fonts for sets (e.g., $\mathcal{S}$). We denote the transpose with a prime (i.e., $\mathbf{A}'$

is the transpose of $\mathbf{A}$). We use arrowed lower-case letters for paths on the graph (e.g., $\boldsymbol{p}$), which are ordered sequences. We use parenthesized superscripts to represent source/target information for the corresponding variables. For example $\boldsymbol{p}^{(s,t)} = \{s = u_0, u_1, ..., u_l = t\}$ is a path from the source node s to the target node t. If the source/target information is clear from the context, we omit the superscript for brevity. A sink node i on the graph is a node without out-links (i.e., $\mathbf{A}(:,i) = 0$). We use subscripts to denote the corresponding variable after setting the nodes indexed by the subscripts as sinks. For example, $\boldsymbol{p}_{\mathcal{I}}^{(s,t)}$ is the path from the source node s to the target node t, which does not go through any nodes indexed by the set $\mathcal{I}$ (i.e., $u_i \notin \mathcal{I}, i = 0, ..., l$). With the above notations, our problems can be formally defined as follows:

Problem 1. **(Pair-Gateway)**

Given: A weighted directed graph $\mathbf{A}$, a source node s, a target node t, and a budget (integer) k;
Find: A set of at most k nodes which has the highest 'gate-way-ness' score wrt s and t.

Problem 2. **(Group-Gateway)**

Given: A weighted directed graph $\mathbf{A}$, a group of source nodes $\mathcal{S}$, a group of target nodes $\mathcal{T}$, and a budget (integer) k;
Find: A set of at most k nodes which has the highest 'gate-way-ness' score wrt $\mathcal{S}$ and $\mathcal{T}$.

In both Problem 1 (Pair-Gateway) and Problem 2 (Group-Gateway), there are two sub-problems: (1) how to define the 'gateway-ness' score of a given subset of nodes $\mathcal{I}$; (2) how to find the subset of nodes with the highest 'gateway-ness' score. In the next two sections, we present the solutions for each, respectively.

3 Proposed 'Gateway-ness' Scores

In this section, we present our definitions for 'Gateway-ness'. We first focus on the case of a single source s and a single target t (Pair-Gateway). We then generalize to the case where both the source and the target are a group of nodes (Group-Gateway)

3.1 Node 'Gateway-ness' Score

Given a single source s and a single target t, we want to measure the 'Gateway-ness' score for a given set of nodes $\mathcal{I}$. We first give the formal definitions in such a setting and then provide some intuitions for our definitions.

For a graph $\mathbf{A}$, we can use random walk with restart to measure the proximity (i.e., relevance/closeness) from the source node s to the target node t, which is defined as follows: Consider a random particle that starts from node s. The particle iteratively transits to its neighbors with probability proportional to the corresponding edge weights. Also at each step, the particle returns to node s with some restart probability $(1-c)$. The proximity score from node s to node t is defined as the steady-state probability $\mathbf{r}(s,t)$ that the particle will be on node t [15]. Intuitively, $\mathbf{r}(s,t)$ is the fraction of time that the particle starting from node s will spend on node t of the graph, after an infinite number of steps.

Intuitively, a set of nodes $\mathcal{I}$ are good gateways wrt s and t if they play an important role in the proximity measure from the source to the target. Therefore, our 'Gateway-ness' score can be defined as follows:

$$\mathrm{g}(s,t,\mathcal{I}) \triangleq \Delta\mathbf{r}(s,t) \triangleq \mathbf{r}(s,t) - \mathbf{r}_{\mathcal{I}}(s,t) \tag{1}$$

where $r_{\mathcal{I}}(s,t)$ is the proximity score from source s to t after setting the subset of nodes indexed by $\mathcal{I}$ as sinks.

3.2 Group 'Gateway-ness' Score

Here we consider the case where the source and/or target consist of more than one nodes. Suppose we have a group of source nodes $\mathcal{S}$ and a group of target nodes $\mathcal{T}$. Then, the 'Gateway-ness' score for a given set of nodes $\mathcal{I}$ can be defined in a similar way:

$$\mathrm{g}(\mathcal{S},\mathcal{T},\mathcal{I}) \triangleq \sum_{s\in\mathcal{S},t\in\mathcal{T}} \Delta\mathbf{r}(s,t) \triangleq \sum_{s\in\mathcal{S},t\in\mathcal{T}} (\mathbf{r}(s,t) - \mathbf{r}_{\mathcal{I}}(s,t)) \tag{2}$$

where $r_{\mathcal{I}}(s,t)$ is the proximity score from s to t by setting the subset of nodes indexed by $\mathcal{I}$ as sinks (i.e., delete all out-edges, by setting $\mathbf{A}(:,i) = 0$ for all $i \in \mathcal{I}$).

4 BASSET: Proposed Fast Solutions

In this section, we address how to quickly find a subset of nodes of the highest 'Gateway-ness' score. We start by showing that the straight-forward methods (referred to as 'Com-RWR') are computationally intractable. Then, we present the proposed BASSET (BASSET-N for Pair-Gateway and BASSET-G for Group-Gateway). For each case, we first present the algorithm and then analyze its effectiveness as well as its computational complexity.

4.1 Computational Challenges

Here, we present the computational challenges and the way we tackle them. For the sake of succinctness, we mainly focus on BASSET-N.

There are two main computational challenges in order to find a subset of nodes with the highest 'Gateway-ness' score. First of all, we need to compute the proximity from the source to the target on different graphs, each of which is a perturbed version of the original graph. This essentially means that we cannot directly apply some powerful pre-computational method to evaluate the proximity from the source to the target (after setting the subset of nodes indexed by $\mathcal{I}$ as sinks). Instead, we have to rely on on-line iterative methods, whose computational complexity is $O(m)$. The challenges are compounded by the need to evaluate $\mathrm{g}(s, t, \mathcal{I})$ (eq. (1)) or $\mathrm{g}(\mathcal{S}, \mathcal{T}, \mathcal{I})$(eq. (2)) an exponential number of times ($\binom{n}{k}$). Putting these together, the straightforward way to find k nodes with the highest 'Gateway-ness' score is $O(\binom{n}{k}m)$. This is computationally intractable. Suppose on a graph with $1,000,000$ nodes, we want to find the best $k = 5$ gateway nodes. If computing each proximity score takes 0.001 seconds, then 2.64×10^{17} years are needed to find the gateways. This is much longer than the age of the universe.[1]

To tackle such challenges, we resort to two main ideas, which are summarized in Theorem 1. According to Theorem 1, in order to evaluate the 'Gateway-ness' score of a given set of nodes, we do not need to actually set these nodes as sinks and compute the proximity score on the new graph. Instead, we can compute it from the original graph. In this way, we can utilize methods based on pre-computation to accelerate the process. Furthermore, since $\mathrm{g}(s, t, \mathcal{I})$ and $\mathrm{g}(\mathcal{S}, \mathcal{T}, \mathcal{I})$ are sub-modular wrt $\mathcal{I}$, we can develop some greedy algorithm to avoid exponential enumeration, and still get some *near-optimal* solution. In Theorem 1, $\mathbf{A}$ is the normalized adjacency matrix of the graph. It is worth pointing out that The proposed methods (BASSET-N and BASSET-G) we will introduce are orthogonal to the specific way of normalization. For simplicity, we use column-normalization throughout this paper. Also, $\mathbf{Q}(\mathcal{I}, \mathcal{I})$ is a $|\mathcal{I}| \times |\mathcal{I}|$ matrix, containing the elements in the matrix $\mathbf{Q}$ which are at the rows/columns indexed by $\mathcal{I}$. Similarly, $\mathbf{Q}(t, \mathcal{I})$ is a row vector with length $|\mathcal{I}|$, containing the elements in the matrix $\mathbf{Q}$ which are at the t^{th} row and the columns indexed by $\mathcal{I}$. $\mathbf{Q}(\mathcal{I}, s)$ is a column vector with length $|\mathcal{I}|$, containing the elements in the matrix $\mathbf{Q}$ which are at the s^{th} column and the rows indexed by $\mathcal{I}$.

Theorem 1. Core Theorem. *Let $\mathbf{A}$ be the normalized adjacency matrix of the graph, and $\mathbf{Q} = (1-c)(\mathbf{I}-c\mathbf{A})^{-1}$. For a given source s and target t, the 'Gateway-ness' score of a subset of nodes $\mathcal{I}$ defined in eq. (1) satisfies the properties P1 and P2. For a given source group $\mathcal{S}$ and target group $\mathcal{T}$, the 'Gateway-ness' score of a subset of nodes $\mathcal{I}$ defined in eq. (2) satisfies the properties P3 and P4, where $s \neq t$, $s, t \notin \mathcal{I}$, $\mathcal{S} \bigcap \mathcal{T} = \emptyset$, $\mathcal{S} \bigcap \mathcal{I} = \emptyset$, and $\mathcal{T} \bigcap \mathcal{I} = \emptyset$.*

P1. $g(s, t, \mathcal{I}) = \mathbf{Q}(t, \mathcal{I})\mathbf{Q}(\mathcal{I}, \mathcal{I})^{-1}\mathbf{Q}(\mathcal{I}, s)$;
P2. $g(s, t, \mathcal{I})$ *is sub-modular wrt the set* $\mathcal{I}$.
P3. $g(\mathcal{S}, \mathcal{T}, \mathcal{I}) = \sum_{s\in\mathcal{S}, t\in\mathcal{T}} \mathbf{Q}(t, \mathcal{I})\mathbf{Q}(\mathcal{I}, \mathcal{I})^{-1}\mathbf{Q}(\mathcal{I}, s)$;
P4. $g(\mathcal{S}, \mathcal{T}, \mathcal{I})$ *is sub-modular wrt the set* $\mathcal{I}$.

Proof of P1: WLOG, we assume that $\mathcal{I} = \{n-k+1, ...n\}$. Let $\mathbf{A}$ and $\tilde{\mathbf{A}}$ be the normalized adjacency matrices of the graph before/after we set the subset of nodes in $\mathcal{I}$ as sinks. Write $\mathbf{A}$ and $\tilde{\mathbf{A}}$ in block form:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{pmatrix}, \; \tilde{\mathbf{A}} = \begin{pmatrix} \tilde{\mathbf{A}}_{1,1} & \tilde{\mathbf{A}}_{1,2} \\ \tilde{\mathbf{A}}_{2,1} & \tilde{\mathbf{A}}_{2,2} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{1,1} & \mathbf{0} \\ \mathbf{A}_{2,1} & \mathbf{0} \end{pmatrix} \tag{3}$$

[1] According to Wikipedia, (`http://en.wikipedia.org/wiki/Age_of_the_universe`), the age of the universe is about 1.4×10^{10} year.

where $\mathbf{0}$ is a matrix with all zero elements.

Let $\tilde{\mathbf{Q}} = (1-c)(\mathbf{I} - c\tilde{\mathbf{A}})^{-1}$. We can also write $\tilde{\mathbf{Q}}$ and $\mathbf{Q}$ in block form:

$$\mathbf{Q} = (1-c)(\mathbf{I} - c\mathbf{A})^{-1} = \begin{pmatrix} \mathbf{Q}_{1,1} & \mathbf{Q}_{1,2} \\ \mathbf{Q}_{2,1} & \mathbf{Q}_{2,2} \end{pmatrix} = (1-c)\begin{pmatrix} \mathbf{I} - c\mathbf{A}_{1,1} & -c\mathbf{A}_{1,2} \\ -c\mathbf{A}_{2,1} & \mathbf{I} - c\mathbf{A}_{2,2} \end{pmatrix}^{-1}$$
$$\tilde{\mathbf{Q}} = \begin{pmatrix} \tilde{\mathbf{Q}}_{1,1} & \tilde{\mathbf{Q}}_{1,2} \\ \tilde{\mathbf{Q}}_{2,1} & \tilde{\mathbf{Q}}_{2,2} \end{pmatrix} = (1-c)\begin{pmatrix} \mathbf{I} - c\mathbf{A}_{1,1} & \mathbf{0} \\ -c\mathbf{A}_{2,1} & \mathbf{I} \end{pmatrix}^{-1}$$

Applying the block matrix inverse lemma [12] to $\tilde{\mathbf{Q}}$ and $\mathbf{Q}$, we get the following equations:

$$\begin{aligned}
\tilde{\mathbf{Q}}_{1,1} &= (1-c)(\mathbf{I} - c\mathbf{A}_{1,1})^{-1}, \quad \tilde{\mathbf{Q}}_{1,2} = \mathbf{0} \\
\tilde{\mathbf{Q}}_{2,1} &= c(1-c)\mathbf{A}_{2,1}(\mathbf{I} - c\mathbf{A}_{1,1})^{-1}, \quad \tilde{\mathbf{Q}}_{2,2} = (1-c)\mathbf{I} \\
\mathbf{Q}_{1,1} &= (1-c)(\mathbf{I} - c\mathbf{A}_{1,1})^{-1} + c^2(\mathbf{I} - c\mathbf{A}_{1,1})^{-1}\mathbf{A}_{1,2}\mathbf{Q}_{2,2}\mathbf{A}_{2,1}(\mathbf{I} - c\mathbf{A}_{1,1})^{-1} \\
\mathbf{Q}_{1,2} &= c(\mathbf{I} - c\mathbf{A}_{1,1})^{-1}\mathbf{A}_{1,2}\mathbf{Q}_{2,2} \\
\mathbf{Q}_{2,1} &= c\mathbf{Q}_{2,2}\mathbf{A}_{2,1}(\mathbf{I} - c\mathbf{A}_{1,1})^{-1}
\end{aligned} \tag{4}$$

Therefore, we have

$$\tilde{\mathbf{Q}}_{1,1} = \mathbf{Q}_{1,1} - \mathbf{Q}_{1,2}\mathbf{Q}_{2,2}^{-1}\mathbf{Q}_{2,1} \tag{5}$$

On the other hand, based on the properties of random walk with restart [15], we have $\mathbf{r}(i,j) = \mathbf{Q}(j,i)$, and $\mathbf{r}_{\mathcal{I}}(i,j) = \tilde{\mathbf{Q}}(j,i)$,$(i,j = 1,...,n)$. Together with eq. (5), we have

$$\mathrm{g}(s,t,\mathcal{I}) = \mathbf{r}(s,t) - \mathbf{r}_{\mathcal{I}}(s,t) = \mathbf{Q}_{1,1}(t,s) - \tilde{\mathbf{Q}}_{1,1}(t,s) = \mathbf{Q}_{1,2}(t,:)\mathbf{Q}_{2,2}^{-1}\mathbf{Q}_{1,2}(:,s) \tag{6}$$

which completes the proofs of P1. $\square$

Proof of P3: Since P1 holds, we have

$$\mathrm{g}(\mathcal{S},\mathcal{T},\mathcal{I}) = \sum_{s\in\mathcal{S},t\in\mathcal{T}} \Delta r(s,t) = \sum_{s\in\mathcal{S},t\in\mathcal{T}} \mathrm{g}(s,t,\mathcal{I}) = \sum_{s\in\mathcal{S},t\in\mathcal{T}} \mathbf{Q}(t,\mathcal{I})\mathbf{Q}(\mathcal{I},\mathcal{I})^{-1}\mathbf{Q}(\mathcal{I},s) \tag{7}$$

which completes the proofs of P3. $\square$

Proof of P2: Let $\mathcal{I},\mathcal{J},\mathcal{K}$ be three subsets and $\mathcal{I} \subseteq \mathcal{J}$. We will first prove by induction that, for any integer power j, the following inequality holds element-wise.

$$\mathbf{A}_{\mathcal{I}}^{j} - \mathbf{A}_{\mathcal{I}\bigcup\mathcal{K}}^{j} \geq \mathbf{A}_{\mathcal{J}}^{j} - \mathbf{A}_{\mathcal{J}\bigcup\mathcal{K}}^{j} \tag{8}$$

It is easy to verify the base case (i.e.,$j=1$) for eq. (8) holds. Next, assume that eq. (8) holds for $j = 1,...,j_0$, and we want to prove that it also holds for the case $j = j_0 + 1$:

$$\begin{aligned}
\mathbf{A}_{\mathcal{I}}^{j_0+1} - \mathbf{A}_{\mathcal{I}\bigcup\mathcal{K}}^{j_0+1} &= \mathbf{A}_{\mathcal{I}}^{j_0+1} - \mathbf{A}_{\mathcal{I}\bigcup\mathcal{K}}^{j_0}\mathbf{A}_{\mathcal{I}} + \mathbf{A}_{\mathcal{I}\bigcup\mathcal{K}}^{j_0}\mathbf{A}_{\mathcal{I}} - \mathbf{A}_{\mathcal{I}\bigcup\mathcal{K}}^{j_0+1} \\
&= (\mathbf{A}_{\mathcal{I}}^{j_0} - \mathbf{A}_{\mathcal{I}\bigcup\mathcal{K}}^{j_0})\mathbf{A}_{\mathcal{I}} + \mathbf{A}_{\mathcal{I}\bigcup\mathcal{K}}^{j_0}(\mathbf{A}_{\mathcal{I}} - \mathbf{A}_{\mathcal{I}\bigcup\mathcal{K}}) \\
&\geq (\mathbf{A}_{\mathcal{J}}^{j_0} - \mathbf{A}_{\mathcal{J}\bigcup\mathcal{K}}^{j_0})\mathbf{A}_{\mathcal{I}} + \mathbf{A}_{\mathcal{I}\bigcup\mathcal{K}}^{j_0}(\mathbf{A}_{\mathcal{J}} - \mathbf{A}_{\mathcal{J}\bigcup\mathcal{K}}) \\
&\geq (\mathbf{A}_{\mathcal{J}}^{j_0} - \mathbf{A}_{\mathcal{J}\bigcup\mathcal{K}}^{j_0})\mathbf{A}_{\mathcal{J}} + \mathbf{A}_{\mathcal{J}\bigcup\mathcal{K}}^{j_0}(\mathbf{A}_{\mathcal{J}} - \mathbf{A}_{\mathcal{J}\bigcup\mathcal{K}}) = \mathbf{A}_{\mathcal{J}}^{j_0+1} - \mathbf{A}_{\mathcal{J}\bigcup\mathcal{K}}^{j_0+1}
\end{aligned} \tag{9}$$

In eq. (9), the first inequality holds because of the induction assumption. The second inequality holds because $\mathbf{A}_{\mathcal{I}} \geq \mathbf{A}_{\mathcal{J}} \geq 0$ holds element-wise, and $\mathbf{A}_{\mathcal{I} \bigcup \mathcal{K}} \geq \mathbf{A}_{\mathcal{J} \bigcup \mathcal{K}} \geq 0$ holds element-wise.

Since $\tilde{\mathbf{Q}} = (1-c)(\mathbf{I} - c\tilde{\mathbf{A}})^{-1} = (1-c)\sum_{j=0}^{\infty}(c\tilde{\mathbf{A}})^j$, we have

$$\begin{aligned} \mathrm{g}(s,t,\mathcal{I} \cup \mathcal{K}) - \mathrm{g}(s,t,\mathcal{I}) &= (1-c)\sum_{j=0}^{\infty}((c\mathbf{A}_{\mathcal{I}})^j(t,s) - (c\mathbf{A}_{\mathcal{I} \bigcup \mathcal{K}})^j(t,s)) \\ &\geq (1-c)\sum_{j=0}^{\infty}((c\mathbf{A}_{\mathcal{J}})^j(t,s) - (c\mathbf{A}_{\mathcal{J} \bigcup \mathcal{K}})^j(t,s)) \\ &= \mathrm{g}(s,t,\mathcal{J} \cup \mathcal{K}) - \mathrm{g}(s,t,\mathcal{J}) \end{aligned} \tag{10}$$

Therefore, $\mathrm{g}(s,t,\mathcal{I})$ is sub-modular, which completes the proof of P2. □

Proof of P4: Since $\mathrm{g}(\mathcal{S},\mathcal{T},\mathcal{I}) = \sum_{s\in\mathcal{S},t\in\mathcal{T}} \mathrm{g}(s,t,\mathcal{I})$ (In other words, $\mathrm{g}(\mathcal{S},\mathcal{T},\mathcal{I})$ is a non-negative linear combination of sub-modular functions) , according to the linearity of sub-modular functions [8], we have that $\mathrm{g}(\mathcal{S},\mathcal{T},\mathcal{I})$ is also sub-modular, which completes the proof of P4. □

4.2 BASSET-N for Problem 1

BASSET-N Algorithm. Our fast solution for Problem 1 is summarized in Alg. 1. In Alg. 1, after initialization (step 1), we first pick a node i_0 with the highest $\frac{\mathbf{r}(s,i)\mathbf{r}(i,t)}{\mathbf{r}(i,i)}$ (step 3). Then, in steps 4-14, we find the rest of the nodes in a greedy way. That is, in each outer loop, we try to find one more node while keeping the current $\mathcal{I}$ unchanged. According to P1 of theorem 1, $\mathbf{v}(i)$ computed in step 7 is the gateway score for the subset $\mathcal{J}$.[2] If the current subset of nodes $\mathcal{I}$ can completely disconnect the source and the target (by setting them as sinks), we will stop the algorithm (step 12). Therefore, Alg. 1 always returns no more than k nodes. It is worth pointing out that in Alg. 1, all the proximity scores are computed from the original graph $\mathbf{A}$. Therefore, we can utilize some powerful methods based on pre-computation to accelerate the whole process. To name a few, for a medium size graph $\mathbf{A}$ (e.g., a few thousands of nodes), we can pre-compute and store the matrix $\mathbf{Q} = (1-c)(\mathbf{I} - c\mathbf{A})^{-1}$; for large unipartite graphs and bipartite graphs, we can use the NB_LIN and BB_LIN algorithms, respectively [15].

Analysis of BASSET-N. In this subsection, we analyze the effectiveness and the efficiency of Alg. 1. First, the effectiveness of the proposed BASSET-N is guaranteed by the following lemma. According to Lemma 1, although BASSET-N is a greedy algorithm, the results it outputs are *near-optimal.*

Lemma 1. Effectiveness of BASSET-N. *Let $\mathcal{I}$ be the subset of nodes selected by Alg. 1 and $|\mathcal{I}| = k_0$. Then, $g(s,t,\mathcal{I}) \geq (1-1/e)max_{|\mathcal{J}|=k_0}\ g(s,t,\mathcal{J})$, where $g(s,t,\mathcal{I})$, and $g(s,t,\mathcal{J})$ are defined by eq.* (1).

Proof: It is easy to verify that the node i_0 selected in step 10 of Alg. 1 satisfies $i_0 = \mathrm{argmax}_{j\notin\mathcal{I},j\neq s,j\neq t}\mathrm{g}(s,t,\mathcal{I} \bigcup j)$. Also, we have $\mathrm{g}(s,t,\phi) = 0$, where ϕ is an

[2] This is because in random walk with restart, we have $\mathbf{r}(i,j) = \mathbf{Q}(j,i)$ for any i,j [15].

Algorithm 1. BASSET-N

```
Input: the normalized adjacency matrix $\mathbf{A}$, the source node $s$, the target node $t$, the
    budget $k$ and the parameter $c$
Output: a set of nodes $\mathcal{I}$, where $|\mathcal{I}| \leq k$.
 1: initialize $\mathcal{I}$ to be empty.
 2: compute the proximity score $\mathbf{r}(s,t)$ from the source node $s$ to the target node $t$.
 3: find $i_0 = \text{argmax}_i \frac{\mathbf{r}(s,i)\mathbf{r}(i,t)}{\mathbf{r}(i,i)}$, where $i = 1, ..., n$ and $i \neq s, i \neq t$. add $i_0$ to $\mathcal{I}$.
 4: for $j = 2$ to $k$ do
 5:   for $i = 1$ to $n$, and $i \neq s, i \neq t$ and $i \notin \mathcal{I}$ do
 6:     let $\mathcal{J} = \mathcal{I} \cup i$.
 7:     compute $\mathbf{v}(i) = \mathbf{r}(\mathcal{J},t)'\mathbf{r}(\mathcal{J},\mathcal{J})^{-1}\mathbf{r}(s,\mathcal{J})'$
 8:   end for
 9:   if $\max_i \mathbf{v}(i) \leq \mathbf{r}(s,t)$ then
10:     find $i_0 = \text{argmax}_i \mathbf{v}(i)$; add $i_0$ to $\mathcal{I}$.
11:   else
12:     break;
13:   end if
14: end for
15: return $\mathcal{I}$
```

empty set. On the other hand, according to Theorem 1, $\text{g}(s,t,\mathcal{I})$ is sub-modular wrt the subset $\mathcal{I}$. Therefore, we have $\text{g}(s,t,\mathcal{I}) \geq (1-1/e)\max_{|\mathcal{J}|=k_0} \text{g}(s,t,\mathcal{J})$, which completes the proof. $\square$

Next, we analyze the efficiency of BASSET-N, which is given in Lemma 2[3]. We can draw the following two conclusions, according to Lemma 2: (1) the proposed BASSET-N achieves a significant speedup over the straight-forward method ($O(n \cdot k^4)$ vs. $O(\binom{n}{k}m)$). For example, in the graph with 100 nodes and 1,000 edges, in order to find the gateway with $k = 5$ nodes, BASSET-N is more than *6 orders of magnitude* faster, and the speedup quickly increases wrt the size of the graph; (2) the proposed BASSET-N is applicable to large graphs since it is linear wrt the number of the nodes.

Lemma 2. Efficiency of BASSET-N. *The computational complexity of Alg. 1 is upper bounded by $O(n \cdot k^4)$.*

Proof: The cost for steps 1-2 is constant. The cost for step 3 is $O(n)$. At each inner loop (steps 6-7), the cost is $O(nj^3 + nj^2)$. The cost for steps 9-13 is $O(n)$. The outer loop has no more than $k-1$ iterations. Putting these together, the computational cost for BASSET-N is:

$$\begin{aligned}\text{Cost(BASSET-N)} &\leq n + \sum_{j=1}^{k}(nj^3 + nj^2 + n) \\ &= n + nk + n\frac{k(k+1)(2k+1)}{6} + n\frac{k^2(k+1)^2}{4} = O(nk^4) \qquad (11)\end{aligned}$$

which completes the proof. $\square$

[3] Here, we assume that the cost to get one proximity score is constant, which can be achieved with pre-computation methods [15].

4.3 BASSET-G for Problem 2

BASSET-G Algorithm. Our fast solution for Problem 2 is summarized in Alg. 2. It works in a similar way as Alg. 1: after initialization (step 1), we first pick a node i_0 with the highest $\sum_{s\in\mathcal{S},\ t\in\mathcal{T}} \frac{\mathbf{r}(s,i)\mathbf{r}(i,t)}{\mathbf{r}(i,i)}$ (step 3). Then, in steps 4-14, we find the rest of the nodes in a greedy way. That is, in each outer-loop, we try to find one more node while keeping the current $\mathcal{I}$ unchanged. If the current subset of the nodes $\mathcal{I}$ can completely disconnect the source group and the target group (by setting them as sinks), we will stop the algorithm (step 10). As in Alg. 1, all the proximity scores are computed from the original graph $\mathbf{A}$. Therefore, we can again utilize those powerful pre-computation based methods to accelerate the whole process.

Algorithm 2. BASSET-G

```
Input: the normalized adjacency matrix $\mathbf{A}$, the source group $\mathcal{S}$, the target group $\mathcal{T}$,
    the budget $k$ and the parameter $c$
Output: a set of nodes $\mathcal{I}$, where $|\mathcal{I}| \leq k$.
 1: initialize $\mathcal{I}$ to be empty.
 2: compute the proximity score $\sum_{s\in\mathcal{S},\ t\in\mathcal{T}} \mathbf{r}(s,t)$ from the source group $\mathcal{S}$ to the target
    group $\mathcal{T}$.
 3: find $i_0 = \text{argmax}_i \sum_{s\in\mathcal{S},\ t\in\mathcal{T}} \frac{\mathbf{r}(s,i)\mathbf{r}(i,t)}{\mathbf{r}(i,i)}$, where $i = 1, ..., n$ and $i \neq s, i \neq t$; add $i_0$
    to $\mathcal{I}$.
 4: for $j = 2$ to $k$ do
 5:   for $i = 1$ to $n$, and $i \neq s, i \neq t$ and $i \notin \mathcal{I}$ do
 6:     let $\mathcal{J} = \mathcal{I} \cup i$.
 7:     compute $\mathbf{v}(i$ as $\mathbf{v}(i) = \sum_{s\in\mathcal{S},\ t\in\mathcal{T}} \mathbf{r}(\mathcal{J},t)'\mathbf{r}(\mathcal{J},\mathcal{J})^{-1}\mathbf{r}(s,\mathcal{J})'$
 8:   end for
 9:   if $\max_i \mathbf{v}(i) \leq \sum_{s\in\mathcal{S},\ t\in\mathcal{T}} \mathbf{r}(s,t)$ then
10:     find $i_0 = \text{argmax}_i \mathbf{v}(i)$; add $i_0$ to $\mathcal{I}$.
11:   else
12:     break;
13:   end if
14: end for
15: return $\mathcal{I}$
```

Analysis of BASSET-G. The effectiveness and efficiency of the proposed BASSET-G are given in Lemma 3 and Lemma 4, respectively. Similar as BASSET-N, the proposed BASSET-G is (1) *near-optimal*; and (2) fast and scalable for large graphs.

Lemma 3. Effectiveness of BASSET-G. *Let $\mathcal{I}$ be the subset of nodes selected by Alg. 2 and $|\mathcal{I}| = k_0$. Then, $g(\mathcal{S},\mathcal{T},\mathcal{I}) \geq (1-1/e)max_{|\mathcal{J}|=k_0}\ g(\mathcal{S},\mathcal{T},\mathcal{J})$, where $g(\mathcal{S},\mathcal{T},\mathcal{I})$, and $g(\mathcal{S},\mathcal{T},\mathcal{J})$ are defined by eq. (2).*

Proof: Similar as for Lemma 1. Omitted for brevity. □

Lemma 4. Efficiency of BASSET-G. *The computational complexity of Alg. 2 is upper bounded by $O(n \cdot (max(k, |\mathcal{S}|, |\mathcal{T}|))^4)$.*

Proof: Similar as for Lemma 2. Omitted for brevity. □

5 Experimental Evaluations

In this section we present experimental results. All the experiments are designed to answer the following questions:

- *Effectiveness:* how effective are the 'Gateway-ness' scores in real graphs?
- *Efficiency:* how fast and scalable are BASSET-N and BASSET-G?

5.1 Data Sets

We used three real data sets, which are summarized in table 2.

Table 2. Summary of the data sets

Name	n	m	**Weight**
PolBooks	105	882	No
AA	418,236	2,753,798	Yes
NetFlix	2,667,199	56,919,190	No

The first data set (*PolBooks*) is a co-purchasing book network.[4] Each node is a political book and there is an edge between two books if purchased by the same person. Overall, we have $n = 105$ nodes and $m = 882$ edges.

The second data set (*AA*) is a co-authorship network, where each node is an author and the edge weight is the number of the co-authored papers between the two corresponding persons. Overall, we have $n = 418,236$ nodes and $m = 2,753,798$ edges.

The last data set (*NetFlix*) is from the Netflix prize[5]. Rows represent users and columns represent movies. If a user has given a particular movie positive ratings (4 or 5), we connect them with an edge. In total, we have 2,667,199 nodes (2,649,429 users and 17,770 movies), and 56,919,190 edges.

5.2 Effectiveness

Here, we evaluate the effectiveness of the proposed 'Gateway-ness' scores. We first compare with several candidate methods in terms of separating the source from the target. And then, we present various case studies.

Quantitative Comparisons. The basic idea of the proposed 'Gateway-ness' scores is to find a subset of nodes which collectively play an important role in measuring the proximity from the source node (or source group) to the target node (or target group). Here, we want to validate this basic assumption. We compare it with the following alternative choices: (a) selecting k nodes with the highest center-piece AND score (CePS-AND) [13]; (b) selecting k nodes with the highest center-piece OR score (CePS-OR) [13]; (c) randomly selecting k

[4] `http://www.orgnet.com/`
[5] `http://www.netflixprize.com/`

Table 3. Effectiveness comparison between BASSET-N and alternatives. x-axis is the budget k and y-axis is the normalized decay of proximity. Higher is better. The proposed BASSET-N is the best.

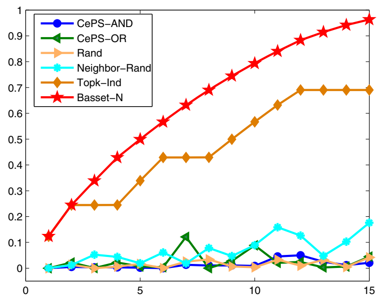


Table 4. BASSET-N on *PolBooks* Graph. ('c' for 'conservative', 'l' for 'liberal', and 'n' for 'neutral')

Node	Book Title	Label
10	Bush country	c
13	Off with their heads	c
103	Back up such up	l
5	Sleeping with the devil	n
8	Ghost wars	n
77	Plan of attack	n
78	Bush at war	c
59	Rise of the vulcanes	c
52	Allies	c
42	The Bushes	c

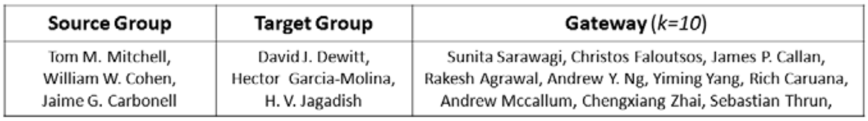
nodes (Rand); (d) randomly selecting k nodes from the neighboring nodes of the source node and the target node (Neighbor-Rand); (e) selecting k nodes with the highest $\frac{\mathbf{r}(s,i)\mathbf{r}(i,t)}{\mathbf{r}(i,i)}$ (Topk-Ind). We randomly select a source node s and a target node t,[6] and then use the different methods to select a subset $\mathcal{I}$ with k nodes. Figure 2 presents the comparison results, where the x-axis is the number of nodes selected (k), and the y-axis is the normalized decay in terms of the proximity score from the source node s to the target node t ($\frac{\mathbf{r}(s,t)-\mathbf{r}_{\mathcal{I}}(s,t)}{\mathbf{r}(s,t)}$). The resulting curves are averaged over 1,000 randomly chosen source-target pairs. From table 3, we can see that (1) the proposed BASSET-N performs best in terms of separating the source from the target; (2)Topk-Ind, where we simply select k nodes with highest $\frac{\mathbf{r}(s,i)\mathbf{r}(i,t)}{\mathbf{r}(i,i)}$, does not perform as well as BASSET-N, where we want to find a subset of k nodes which *collectively* has the highest score $\mathbf{r}(\mathcal{I},t)'\mathbf{r}(\mathcal{I},\mathcal{I})^{-1}\mathbf{r}(s,\mathcal{I})'$.

Case Studies. Next, we will show some case studies, to demonstrate the effectiveness of BASSET-N and BASSET-G.

PolBooks. For this data set, the nodes are political books and the existence of the edge indicates the co-purchasing (by the same person) of the two books. Each book is annotated by one of the following three labels: 'liberal', 'conservative' and 'neutral'. We pick a 'liberal' book ('The Price of Loyalty') as the source node, and a 'conservative' book ('Losing Bin Laden') as the target node. Then, we ran the proposed BASSET-N to find the gateway with 10 nodes. The result is presented in table 4. The result is again consistent with human intuition, - the resulting gateway books are either popular books in one of the two communities ('conservative' vs. 'liberal') such as, 'Bush country' from 'conservative', 'Back up suck up' from 'liberal', etc; or those 'neutral' books which are likely to be purchased by readers from both communities (e.g., 'Sleeping with the devil', etc).

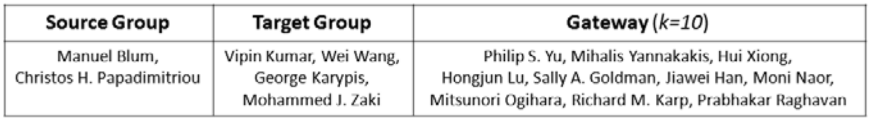
[6] The result when source and target are a group of nodes is similar, and omitted for brevity.

Table 5. BASSET-G on AA Network



Source Group	Target Group	Gateway ($k=10$)
Tom M. Mitchell, William W. Cohen, Jaime G. Carbonell	David J. Dewitt, Hector Garcia-Molina, H. V. Jagadish	Sunita Sarawagi, Christos Faloutsos, James P. Callan, Rakesh Agrawal, Andrew Y. Ng, Yiming Yang, Rich Caruana, Andrew Mccallum, Chengxiang Zhai, Sebastian Thrun,

(a) A group of people in 'text' to a group of people in 'databases'



Source Group	Target Group	Gateway ($k=10$)
Manuel Blum, Christos H. Papadimitriou	Vipin Kumar, Wei Wang, George Karypis, Mohammed J. Zaki	Philip S. Yu, Mihalis Yannakakis, Hui Xiong, Hongjun Lu, Sally A. Goldman, Jiawei Han, Moni Naor, Mitsunori Ogihara, Richard M. Karp, Prabhakar Raghavan

(b) A group of people in 'theory ' to a group of people in 'bioinfomatics'

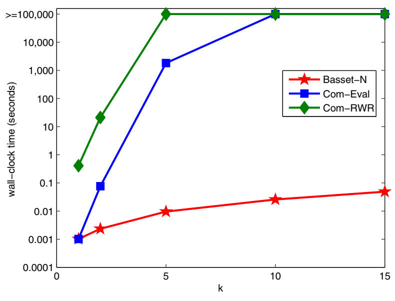
AA. We use this data set to perform case studies for the proposed BASSET-G. We choose (1) a group of people from a certain field (e.g., 'text', 'theory', etc) as the source group $\mathcal{S}$; and (2) another group of people in some other field (e.g., 'databases', 'bioinfomatics', etc) as the target group $\mathcal{T}$. Then, we ran the proposed BASSET-N to find the gateway with $k = 10$ nodes. Table 5 lists some results. They are all consistent with human intuition, - the resulting authors are either productive authors in one of the two fields, or multi-disciplinary, who have close collaborations to both the source and the target groups of authors.

5.3 Efficiency

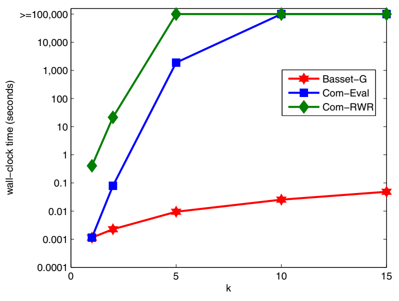
We will study the wall-clock running time of the proposed BASSET-N and BASSET-G here. Basically, we want to answer the following two questions:

1. *(Speed)* What is the speedup of the proposed BASSET-N and BASSET-G over the straightforward methods?
2. *(Scalability)* How do BASSET-N and BASSET-G scale with the size of the graph (n and m)?

First, we compare BASSET-N and BASSET-G with two straightforward methods: (1) 'Com-RWR', where we use combinatorial enumeration to find the gateway and, for each enumeration, we compute the proximity from the *new* graph; and (2) 'Com-Eval', where we use combinatorial enumeration to find the gateway, and for each enumeration, we compute the proximity from the *original* graph. Figure 1 shows the comparison on *PolBooks* graph. We can draw the following conclusions. (1) Straightforward methods ('Com-RWR' and 'Com-Eval') are computationally intractable even for a small graph. For example, it takes more than $1,000$ seconds and $100,000$ seconds to find the $k = 5$ gateway by 'Com-Eval' and by 'Com-RWR', respectively. (2) The speedup of the proposed BASSET-N and BASSET-G over both 'Com-Eval' and 'Com-RWR' is significant - in most cases, we achieve *several (up to 6) orders of magnitude* speedups. (3) The speedup of the proposed BASSET-N and BASSET-G over both 'Com-RWR' and 'Com-Eval' quickly increases wrt the size of the gateway k. Note that we stop running the program if it takes more than 100,000 seconds (i.e., longer than a day).

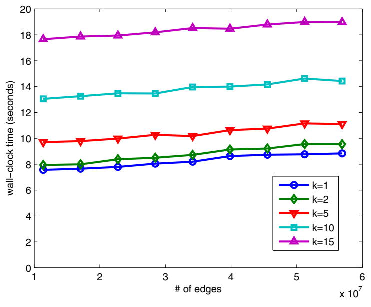


(a) BASSET-N

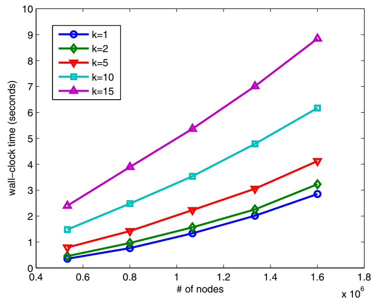


(b) BASSET-G

Fig. 1. Comparison of speed on *PolBooks* graph. Wall-clock time vs. k. Lower is better. Time is in logarithm scale. The proposed BASSET-N and BASSET-G (red star) are significantly faster.

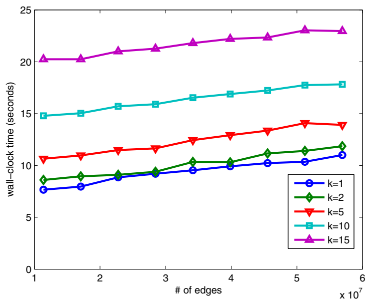


wall-clock time vs. m
(fix $n = 2,667,199$)

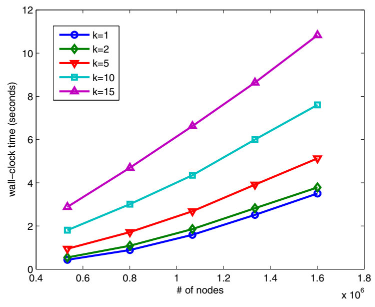


wall-clock time vs. n
(fix $m = 23,874,700$)

(a) BASSET-N



wall-clock time vs. m
(fix $n = 2,667,199$)



wall-clock time vs. n
(fix $m = 23,874,700$)

(b) BASSET-G

Fig. 2. Scalability of BASSET. Wall-clock time vs. the size of the graph. Lower is better. $|\mathcal{S}| = |\mathcal{T}| = 5$.

Next, we evaluate the scalability of the proposed BASSET-N and BASSET-G wrt the size of the graph, using the largest data set (*NetFlix*). From figure 2, we can make the following conclusions: (1) if we fix the number of nodes (n) in the graph, the wall-clock time of both BASSET-N and BASSET-G is almost *constant* wrt the number of edges (m); and (2) if we fix the number of edges (m) in the graph, the wall-clock time of both BASSET-N and BASSET-G is *linear* wrt the number of nodes (n). Therefore, they are suitable for large graphs.

6 Related Work

In this section, we review the related work, which can be categorized into three parts:

Betweenness centrality. The proposed 'Gateway-ness' scores relate to measures of betweenness centrality, both those based on the shortest path [4], as well as those based on random walk [10]. When the gateway set size is $k = 1$, the proposed 'Gateway-ness' scores can be viewed as *query-specific* betweenness centrality measures. Moreover, in the proposed BASSET-N and BASSET-G, we aim to find a subset of nodes *collectively*, wherein traditional betweenness centrality, we usually calculate the score for each node *independently* (and then might pick k nodes with the highest individual scores).

Connection subgraphs. In the proposed BASSET-N, the idea of finding a subset of nodes wrt the source/target is also related to the concept of connection subgraphs, such as [3,7,13]. However, in connection subgraphs, we aim to find a subset of nodes which have *strong* connections among themselves for the purpose of visualization. While in the proposed BASSET-N, we implicity encourage the resulting subset of nodes to be disconnected with each other so that they are able to *collectively disconnect* the target node from the source node to the largest extent (if we set them as sinks).

Graph proximity. The basic idea of the proposed BASSET-N and BASSET-G is to find a subset of nodes which will bring the largest decrease of the proximity score from the source node (or the source group) to the target node (or the target group). Graph proximity itself is an important building block in many graph mining settings. Representative work includes the BANKS system [1], link prediction [9], content-based image retrieval [6], cross-modal correlation discovery [11], pattern matching [14], ObjectRank [2], RelationalRank [5], etc.

7 Conclusion

In this paper, we study how to find good 'gateway' nodes in a graph, given one or more source and target nodes. Our main contributions are: (a) we formulate the problem precisely; (b) we develop BASSET-N and BASSET-G, two fast (up to *6,000,000x* speedup) and scalable (*linear* wrt the number of the nodes in the graph) algorithms to solve it in a provably near-optimal fashion, using sub-modularity. We applied the proposed BASSET-N and BASSET-G on real data sets to validate the effectiveness and efficiency.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grants No. DBI-0640543 IIS-0705359 CNS-0721736 IIS0808661 iCAST and was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract No. DE-AC52-07NA27344. This work is also partially supported by an IBM Faculty Award, a SPRINT gift, with additional funding from Intel, and Hewlett-Packard. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

References

1. Aditya, B., Bhalotia, G., Chakrabarti, S., Hulgeri, A., Nakhe, C., Parag, S.S.: Banks: Browsing and keyword searching in relational databases. In: VLDB, pp. 1083–1086 (2002)
2. Balmin, A., Hristidis, V., Papakonstantinou, Y.: Objectrank: Authority-based keyword search in databases. In: VLDB, pp. 564–575 (2004)
3. Faloutsos, C., McCurley, K.S., Tomkins, A.: Fast discovery of connection subgraphs. In: KDD, pp. 118–127 (2004)
4. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry, 35–41 (1977)
5. Geerts, F., Mannila, H., Terzi, E.: Relational link-based ranking. In: VLDB, pp. 552–563 (2004)
6. He, J., Li, M., Zhang, H.-J., Tong, H., Zhang, C.: Manifold-ranking based image retrieval. ACM Multimedia, 9–16 (2004)
7. Koren, Y., North, S.C., Volinsky, C.: Measuring and extracting proximity in networks. In: KDD, pp. 245–255 (2006)
8. Krause, A., Guestrin, C.: Near-optimal nonmyopic value of information in graphical models. In: UAI, pp. 324–331 (2005)
9. Liben-Nowell, D., Kleinberg, J.: The link prediction problem for social networks. In: Proc. CIKM (2003)
10. Newman, M.: A measure of betweenness centrality based on random walks. Social Networks 27, 39–54 (2005)
11. Pan, J.-Y., Yang, H.-J., Faloutsos, C., Duygulu, P.: Automatic multimedia cross-modal correlation discovery. In: KDD, pp. 653–658 (2004)
12. Piegorsch, W., Casella, G.E.: Inverting a sum of matrices. SIAM Review 32, 470 (1990)
13. Tong, H., Faloutsos, C.: Center-piece subgraphs: problem definition and fast solutions. In: KDD, pp. 404–413 (2006)
14. Tong, H., Faloutsos, C., Gallagher, B., Eliassi-Rad, T.: Fast best-effort pattern matching in large attributed graphs. In: KDD, pp. 737–746 (2007)
15. Tong, H., Faloutsos, C., Pan, J.-Y.: Random walk with restart: Fast solutions and applications. Knowledge and Information Systems: An International Journal, KAIS (2008)

Ensemble Learning Based on Multi-Task Class Labels

Qing Wang[1,2] and Liang Zhang[1]

[1] School of Computer Science, Fudan University, Shanghai, China
{wangqing,lzhang}@fudan.edu.cn
[2] School of Management Science and Engineering, Anhui University of Technology, Ma'anshan, China

Abstract. It is well known that diversity among component classifiers is crucial for constructing a strong ensemble. Most existing ensemble methods achieve this goal through resampling the training instances or input features. Inspired by MTForest and AODE that enumerate each input attribute together with the class attribute to create different component classifiers in the ensemble. In this paper, we propose a novel general ensemble method based on manipulating the class labels. It generates different biased new class labels through the Cartesian product of the class attribute and each input attribute, and then builds a component classifier for each of them. Extensive experiments, using decision tree and naive Bayes as base classifier respectively, show that the accuracy of our method is comparable to state-of-the-art ensemble methods. Finally, the bias-variance decomposition results reveal that the success of our method mainly lies in that it can significantly reduce the bias of the base learner.

1 Introduction

Ensemble learning algorithms train multiple base learners and then combine their predictions to make final decision. Since the generalization ability of an ensemble could be significantly better than that of a single learner, studying the methods for constructing good ensembles has attracted a lot of attentions in machine learning literature during the past decade [17]. Generally, the design of a classifier ensemble contains two subsequent steps, i.e. constructing multiple component classifiers and then combining their predictions.

It is well-recognized that in order to get a strong ensemble, the component classifiers should be with high accuracy as well as high diversity [9,22]. Most existing ensemble methods achieve this goal through resampling the training instances or input features. Among them, Bagging [3], Boosting [10], and Random Subspace [14] are three general techniques widely used in many applications. Both Bagging and Boosting train base classifiers by resampling training instances, while Random Subspace trains base classifiers by using different random subsets of input features.

Another general method for constructing classifier ensemble is to manipulate the class labels that given to the base classifier, i.e. by transforming the learning problem into a collection of related learning problems that use the same input instances but different assignment of the class labels. Most existing methods of this type achieve this by converting the given learning problem into a collection of different but related binary learning problems and then learning a component classifier for each of them. Representative examples include Error-Correcting Output Codes (ECOC)[8] and pairwise ensemble(also known as round robin ensemble) [12], etc. The main deficiency of existing ensemble methods based on manipulating the class labels lies in that, first, they can only be applied to multi-class learning problem; second, although they often can improve the classification accuracy of the base classifiers, their performance gain often is not as large as AdaBoost [12].

Inspired by MTForest[19] and AODE[20] that enumerate each input attribute together with the class attribute to create different component classifiers in the ensemble. In this paper, we propose a novel way to manipulate the class labels for constructing strong ensemble of classifiers by generating different biased new class labels through the Cartesian product of class attribute and each input attribute. This method can be applied to both binary and multi-class learning problems. Extensive experiments show that its performance is superior to AdaBoost. Bias-variance decomposition shows that, the success of it mainly owe to its ability to significantly reduce the bias of the base learner.

The rest of this paper is organized as follows. In section 2, we introduce the background and give a brief review on related work. In section 3, we propose the MACLEN (Multi-tAsk Class Labels based ENsemble) method. Then we report our experimental results in section 4. Finally, we conclude the paper in section 5.

2 Background and Related Work

2.1 Multi-Task Learning

Multi-Task Learning (MTL) trains multiple tasks simultaneously while using a shared representation and has been the focus of much interest in machine learning community over the past decade. It has been empirically [5] as well as theoretically [1] shown can often significantly improve performance relative to learning each task independently. When the training signals are for multiple tasks other than the main task, from the point of view of the main task, the other tasks are serving as a bias [5]. This multi-task bias causes the learner to prefer hypotheses that can explain more than one task, i.e. it must be biased to prefer hypotheses that have utility across these multiple tasks.

In most machine learning applications, however, we are only given the training data which is composed of input attributes and class attribute (main task) and we do not have any other related tasks information. So how to derive related tasks from the given data is crucial for utilizing MTL paradigm to improve the generalization performance. It has been shown in [6] that some of the attributes that attribute selection process discards can beneficially be used as extra related tasks for inductive bias transfer.

2.2 MTForest

Because in multi-task learning extra task is served as additional inductive bias, an ensemble can be constructed by using different extra task to bias each component learner in the ensemble so as to generate different component learners [19]. The multi-task learning theory reveals that the component learner will often be with higher accuracy if the extra task is related to the main task and the component learner will be with diversity if the each extra task represents different bias.

Inspired by this, we propose the MTForest [19] algorithm which enumerates each input attribute as extra task to introduce different additional inductive bias to generate component decision trees in the ensemble. In MTForest, the construction of each component decision trees is related to both the class attribute and the given input attribute. The learning process is similar to standard C4.5 decision tree learning algorithm except that the Information Gain criteria of each split S_i is calculated by combine the class attribute and the given input attribute, showing below:

$$\text{MTIG}(S_i) = \text{ClassAttributeIG}(S_i) + weight * \text{InputAttributeIG}(S_i)$$

The prediction of the ensemble is produced by aggregating the predictions of all these component decision trees. Experimental results show that MTForest can achieve significant improvement over Bagging and is robust to noise.

2.3 Averaged One-Dependence Estimators

Naive Bayes is an efficient and effective learning algorithm. Denote an instance x as a vector $\langle a_1, a_2, ..., a_n\rangle$, where a_i is the value of i-th attribute, and c is the class value . Naive Bayes simply assumes that the attributes are independent given the class, i.e.

$$P(x|c) = \prod_{i=1}^{n} P(a_i|c)$$

The conditional independence assumption unable to capture important attribute dependence information which sometimes hamper the performance of it seriously. So it needs to relax the assumption effectively to improve its classification performance.

In last fewer years, numerous semi-naive Bayesian classifiers that try to exploit attribute dependencies in moderate orders have been proposed and demonstrate improved performance on naive Bayes. Among those classifiers, approaches that utilize ODE(One-Dependence Estimator) have demonstrated remarkable performance [11,15,18,20]. Representative examples include TAN and SPTAN[11], HNB[15], SNODE[18] AODE and its variants, etc. ODE paradigm learning restricts that each attribute can only depend on one parent in addition to the class, and thus it follows that

$$P(x|c) = \prod_{i=1}^{n} P(a_i|pa(i), c)$$

where $pa(i)$ denotes the parent attribute of the i-th attribute, and is determined in the structure learning process.

Averaged One-Dependence Estimators(AODE) [20] is the ensemble technique of utilizing ODE. For simplicity and to avoid structure learning, AODE built an ODE for each attribute, in which the attribute is simply set to be the parent of all other attributes. The final prediction is produced by aggregating the predictions of all these ODEs. A lot of empirical study show that AODE can achieve significant improvement over naive Bayes and AdaBoost [15,20].

3 The MACLEN Method

The success of MTForest and AODE show that enumerating each input attribute together with the class attribute to create different component classifier is a powerful way to construct strong ensemble. However, MTForest and AODE are specifically for ensembling decision tree and naive bayes, respectively. Both of them need to revise the base classifiers to incorporate the information from the given input attribute. While most general ensemble method, such as Bagging, AdaBoost, Random Subspace, is transparent to the base classifier and can easily be applied to any base classifier. Therefore, it will be interesting to ask whether there exists another way to incorporate the information of the input attribute into the base classifier while is transparent to the base classifier.

In this and subsequent sections, we will argue and show that this can be achieved based on generating new class labels through the Cartesian product of the input attribute and class attribute. Since every new class label contains the information of both the class and input attribute, we call it multi-task class label and the ensemble method as MACLEN (Multi-tAsk Class Labels based ENsemble).

3.1 MACLEN: Algorithm Definition

In this subsection, we present the MACLEN method in detail. Assume that the instance in the training data set is represented by $n + 1$-dimension vector $\langle A_1, A_2, ..., A_n, C\rangle$ where A_i is the i-th input attribute and C is the class attribute. An instance x is represented by a vector $\langle a_1, a_2, ..., a_n, c\rangle$, where a_i is the value of attribute A_i and c is the value of class attribute C respectively. For the sake of simplicity, in this paper, we suppose that all the input attributes are discrete.

In MACLEN (see Algorithm 1), an ensemble is constructed by using each input attribute together with the class attribute to generate different but related learning problems independently. When given an input attribute A_i, a new attribute C_i^* is constructed whose values are the Cartesian product of class attribute and this attribute. We then create a new data representation from the original input data by removing this attribute and class attribute and setting the new attribute C_i^* as the class attribute, i.e. the new data representation is $\langle A_1, ..., A_{i-1}, A_{i+1}, ..., A_n, C_i^*\rangle$. Next, each instance $x : \langle a_1, a_2, ..., a_n, c\rangle$ in the original input is transformed into the new data representation by deleting the value a_i and setting the class value as ca_i, i.e. the new transformed instance is $\langle a_1, ..., a_{i-1}, a_{i+1} ..., a_n, ca_i\rangle$. Finally, this new relabeled data set is given to the base learning algorithm, which constructs a classifier h_i. By enumerating each input attribute, we obtain an ensemble of n component classifiers $\{h_1, h_2, ... , h_n\}$.

To classify an unlabeled instance, $x : \langle a_1, a_2, ..., a_n\rangle$, we employ the following method. Each component classifier h_i in the ensemble provides probabilities for the class membership of x while the possible class labels is C_i^*. Since the probability output of h_i can be seen as the joint probability distribution of class attribute C and attribute A_i, we can derive the probability of x belonging to original class label c using the *marginal probability*, i.e.:

$$\hat{P}_{h_i}(c|x) = \sum_{a \in A_i} \hat{P}^*_{h_i}(ca|x) \tag{1}$$

Algorithm 1. MACLEN

```
Input: BaseClassifier - the base learning algorithm.
       $T$ - training data set in which instance is represented by $n+1$-dimension vector
       $\langle A_1, A_2, ..., A_n, C\rangle$.
Output: A ensemble of base classifiers $E$
  $E$={};
  for each attribute $A_i$ in the input
    1. Construct a new attribute $C_i^*$ whose values are the Cartesian product of class attribute $C$
       and attribute$A_i$.
    2. Generate a new data representation from the original input by removing attribute $A_i$ and
       class attribute $C$ and setting the new attribute $C_i^*$ as class attribute.
    3. Generate a new data set $T^*$ by transforming each original input instance $x$ in $T$ into the
       new data representation.
    4. $h_i$=BaseClassifier$(T^*)$.
    5. $E$=$E \bigcup h_i$.
  return $E$
```

where $\hat{P}_{h_i}(c|x)$ is the estimated probability of instance *x* belonging to original class label *c* according to classifier h_i, and $\hat{P}^*_{h_i}(ca|x)$ is the estimated probability of *x* belonging to new class label ca according to classifier h_i. Furthermore, if the value of attribute A_i is given (denoted as a_i), we can also derive the probability of *x* belonging to original class label *c* using the *conditional probability*, i.e.:

$$\hat{P}_{h_i}(c|x) = \frac{\hat{P}^*_{h_i}(ca_i|x)}{\sum_{y\in C}\hat{P}^*_{h_i}(ya_i|x)} \tag{2}$$

Compared to marginal probability (Eq.1), the conditional probability (Eq.2) can utilize extra information about the value of attribute A_i. So, in our implementation, we first choose Equation (2) to calculate the class membership probabilities of each component classifier, only when the Equation (2) is undefined [1], we then turn to use Equation (1) to calculate the class membership probabilities.

Then we compute the class membership probabilities for the entire ensemble as:

$$\hat{P}(c|x) = \frac{\sum_{i=1}^{n}(\hat{P}_{h_i}(c|x))}{n} \tag{3}$$

where $\hat{P}(c|x)$ is the probability of *x* belonging to class *c*. At last, we select the most probable class as the label for *x*, i.e. $E(x) = \arg\max_{c\in C}\hat{P}(c|x)$.

It is noteworthy that for MTCLAN, it enumerates each input attribute to generate each component classifiers in the ensemble, hence its ensemble size is equal to the number of input attributes which is same as MTForest and AODE but is different from most other ensemble methods such as Bagging and Boosting that need to specify the ensemble size. In addition, the building process of each component classifiers do not depend on each other, so MACLEN can easily be parallelized.

[1] The denominator of Equation (2) is 0 (For example, when decision tree is chosen as base classifier and some leafs do not contain revelent labels) or the value of attribute A_i is missing.

To illustrate this algorithm, consider the learning problem described in UCI data set *adult* [2]. In this problem, we are given 14 attributes representing 14 different aspect information of a person, such as the *age*, *martial-status*, *workclass*, *education*, *occupation*, *race*, *sex*, and so on, and the class attribute indicating whether the annual income of this person is larger than 50*k* or not. Our goal is to build a predictive model from the labeled training data which can be used to predict the income type of an unlabeled person given the values of these 14 attributes. In MACLEN, we use each of these 14 input attributes to generate different but related learning problems. For example, if attribute *sex* whose values are $\{male, female\}$ is chosen, we construct a new attribute as the class attribute whose values are $\{male :> 50k, male :\leq 50k, female :> 50k, female :\leq 50k\}$. This new learning problem is different from the original problem because it not only has to predict the income type of the person, but also the sex type of the person. Then we can construct a base classifiers h_{sex}. Assume that when to classify an unlabeled example and the given *sex* value is *female*, we can derive the probabilities for the original class membership $> 50k$ and $\leq 50k$ only from the predicted probabilities of new class membership $female :> 50k$ and $female :\leq 50k$ in h_{sex}. Through using different attribute we construct different learning problems which have different class labels, while these learning algorithms are also related to each other since they all have to predict the income type of the person.

3.2 Discussion

In MACLEN, we only deal with discrete input attributes. For numeric input attribute, some discretize method must be used ahead. Since discretize methods based on different split criteria often result in different number of discrete values and different intervals for the same numerical attribute, this provides a way to build several different component classifiers from a single numerical attribute which can be used to enlarge the ensemble size of the MACLEN.

Note that the existing of discrete attributes which have large number of values may hamper the performance of MACLEN. These attributes make the corresponding new class label set very large. Since the given training instance number is fixed, the instance-to-class ratio will decrease dramatically, which likely to result in inaccurate component classifier. So it may be better to cluster the values of these attributes into small number of groups respectively, and assign all the values in the same group the same discrete value. By using different cluster method, we can also build several different component classifiers from a single discrete attribute.

4 Experiments and Results

4.1 Experimental Setup

We conduct experiments under the framework of *Weka*[21]. For the purpose of our study, we use the 30 well-recognized data sets from the UCI repositories [2] which represent a wide range of domains and data characteristics. A brief description of these data sets is shown in Table 1. We adopted the following three steps to preprocess each

Table 1. Description of the data sets used in the experiments

Datasets	Size	Attribute	Classes	Datasets	Size	Attribute	Classes
adult	48842	15	2	ionosphere	351	35	2
anneal	898	39	6	iris	150	5	3
autos	205	26	7	kr-vs-kp	3196	37	2
balance-scale	625	5	3	letter	20000	17	26
breast-cancer	286	10	2	mushroom	8124	23	2
car	1728	7	4	nursery	12960	9	5
colic	368	23	2	primary-tumor	339	18	21
credit-a	690	16	2	segment	2310	20	7
credit-g	1000	21	2	sick	3772	30	2
diabetes	768	9	2	soybean	683	36	19
glass	214	10	7	vehicle	846	19	4
heart-c	303	14	5	vowel	990	14	11
heart-h	294	14	5	waveform	5000	41	3
hepatitis	155	20	2	yeast	1484	10	10
hypothyroid	3772	30	4	zoo	101	18	7

data set. First, missing values in each data set are filled in using the unsupervised filter *ReplaceMissingValues* in *Weka*; Second, numeric attributes are discretized using the supervised filter *Discretize* in *Weka* which use the MDL discretization method; Third, we use the unsupervised filter *Remove* in *Weka* to delete attributes that do not provide any information to the class, two occurred within the 30 data sets, namely *Sequence Name* in data set *yeast* and *Animal* in data set *zoo*.

In our experiments, we compare MACLEN to Bagging, Boosting and Random Forest when using C4.5 as base classifier. It is well recognized that Bagging is often unable to enhance the performance of stable classifiers. So we compare MACLEN to Boosting and AODE when using naive Bayes as base classifier. And we study MTCLAN for naive Bayes using the Laplace estimation and M-estimation to smooth probability esitmation which is denoted as MACLAN(L) and MACLAN(M) , respectively. The C4.5, Bagging, Boosting (we use the multi-class version AdaBoost.M1 [10]), Random Forest, Naive Bayes and AODE are all already implemented in *Weka*, so we only implement our algorithm under the framework of *Weka*. We set the ensemble size as 50 for all compared methods and keep other parameters at their default values in Weka. It is noteworthy that for our method, the ensemble size is just the number of input attributes which is often far smaller than 50 on these data sets. The classification accuracy and standard deviation of each algorithm on a data set was obtained via 10 runs of ten-fold cross validation. Runs with the various algorithms were carried out on the same training sets and evaluated on the same test sets.

To compare two learning algorithms across all these domains, we first adopt the widely used pairwise two-tailed *t*-test with 95% confidence level to compare our algorithm with other algorithms. Recently, it has been proposed that the best method to compare multiple algorithms over multiple data sets is to compare their overall ranks [7]. So, we also present the overall ranks of our algorithm and other ensemble methods compared. Note that the smaller the rank, the better the method.

4.2 Experimental Results

Table 2 shows the detailed experimental results of the mean classification accuracy and standard deviation of C4.5, Random Forest and three ensemble methods using C4.5 as base classifier on each data set. And the mean values, overall ranks and the pairwise *t*-test results are summarized at the bottom of the table. From Table 2 we can see that MACLEN can achieve substantial improvement over C4.5 on most data set (11 wins and 3 losses) which suggests that MACLEN is potentially a good ensemble technique for decision trees. MACLEN can also gain significantly improvement over Bagging (7 wins and 2 losses) and is comparable to two state-of-the-art ensemble technique for decision trees, AdaBoost (6 wins and 4 losses) and RandomForest (4 wins and 4 losses). The overall rank of MACLEN on these 30 data sets is 2.22 which the smallest among all these ensemble methods.

Table 3 shows the detailed experimental results of the mean classification accuracy and standard deviation of naive Bayes, AODE, MACLAN(L) and MACLAN(M) using naive Bayes as base classifier on each data set. The mean values, overall ranks and the pairwise *t*-test results are also summarized at the bottom of the table. From Table 3, we can see that MACLEN(M) is significantly better than MACLEN(L). This is very likely due to that, compared to original problem, the number of instance belong to each new class label is smaller and the number of class labels is larger. Thus , compared to Laplace estimation, the M-estimation which places more emphasis on data than prior could make probability estimation effectively in this situation. MACLEN can achieve substantial improvement (12 wins and 1 loss, 16 wins and 1 loss, respectively) over naive Bayes and gain improvement over AdaBoost (8 wins and 5 losses, 9 wins and 5 loss, respectvely). Furthermore, MACLEN(M) is comparable to AODE which is the state-of-the-art ensemble method for naive Bayes. Note that MACLEN is a general ensemble method while AODE not.

We also test our method on these 30 UCI data sets under artificial noise in the class labels to study its robustness. Following the method in [4], the noisy version of each training data set is generated by choosing 5% instances and changing their class labels to other incorrect labels randomly. Due to space limited, we do not list the detailed results of the accuracy and standard deviation on each data set here. The experimental results show that MACLEN can significantly outperform AdaBoost (12 wins 1 loss for C4.5 as base classifier, 13 wins 2 losses for naive Bayes as base classifier)in this situation, and is comparable to RandomForest and AODE respectively.

4.3 Bias-Variance Decomposition

To understand the working mechanism of MACLEN, we use the bias-variance decomposition to analysis it. The bias-variance decomposition is a powerful tool for investigating the working mechanism of learning algorithms. Given a learning target and the size of training set, it breaks the expected error of a learning approach into the sum of three non-negative quantities, i.e. the intrinsic noise, the bias, and the variance. At present there exist several kinds of bias-variance decomposition schemes [13]. In this paper, we adopt the one proposed by [16] which has already been implemented in *weka*.

The training data are divided into training and test sets each containing half the data. 50 local training sets are sampled from the training set, each local set containing 50% of

Table 2. Accuracy and standard deviation of C4.5, Bagging, AdaBoost, Random Forest and MACLEN on the 30 UCI data sets. Bottom rows of the table present the mean values, overall ranks and Win-Loss-Tie (*w/l/t*) comparisons between MACLEN against other algorithms using pairwise *t*-tests at 95% significance level, respectively.

Data sets	C4.5	Bagging	AdaBoost	Random Forest	MACLEN
adult	86.74±0.39	86.94±0.36	85.49±0.40	85.59±0.40	85.79±0.37
anneal	98.78±0.91	98.79±0.87	99.61±0.62	99.42±0.84	99.19±0.82
autos	76.39±9.55	83.89±8.51	86.38±6.94	87.06±6.86	87.02±7.64
balance-scale	69.32±3.89	69.26±3.81	69.31±3.90	69.01±3.84	69.39±3.72
breast-cancer	75.26±5.04	73.76±5.85	66.04±8.21	70.37±7.34	67.82±7.22
car	92.22±2.01	93.59±1.79	96.72±1.50	94.42±1.54	94.43±1.71
colic	84.72±5.94	85.10±5.68	79.48±6.36	73.29±5.52	84.48±5.26
credit-a	86.58±3.53	86.17±3.56	82.72±4.36	84.36±4.01	84.71±3.98
credit-g	72.17±3.49	73.66±3.69	71.69±3.98	73.76±3.63	72.59±3.44
diabetes	77.34±4.91	77.33±4.72	77.16±4.38	76.59±4.78	77.19±4.60
glass	75.23±9.46	77.10±9.13	75.28±9.41	76.94±8.07	76.99±9.21
heart-c	77.32±6.20	80.41±6.38	79.57±6.57	80.78±5.83	80.23±5.61
heart-h	80.96±6.91	80.04±7.25	82.34±6.32	81.84±6.35	81.31±6.97
hepatitis	81.32±9.48	83.26±10.18	83.61±8.93	85.52±8.03	83.94±9.74
hypothyroid	99.28±0.42	99.31±0.41	99.50±0.37	99.27±0.40	99.44±0.38
ionosphere	89.49±5.12	91.03±5.33	93.11±4.08	92.99±3.88	91.77±4.49
iris	93.87±4.89	94.47±5.02	94.61±5.42	94.19±5.81	94.87±5.35
kr-vs-kp	99.44±0.37	99.46±0.37	99.60±0.31	99.23±0.46	99.54±0.36
letter	78.75±0.78	81.87±0.95	89.94±0.75	92.01±0.61	90.72±0.68
mushroom	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00	100.0±0.00
nursery	97.18±0.46	97.42±0.43	99.75±0.17	99.06±0.33	98.85±0.35
primary-tumor	41.01±6.59	45.19±6.16	42.63±6.61	41.27±6.09	43.16±6.52
segment	95.23±1.37	95.67±1.22	96.51±1.22	96.91±1.05	97.02±1.01
sick	97.82±0.75	97.84±0.76	97.63±0.78	97.70±0.73	97.81±0.76
soybean	92.63±2.71	94.05±2.61	94.04±2.61	93.73±2.69	93.44±2.73
vehicle	70.77±3.86	70.84±4.01	72.53±3.95	73.25±3.72	72.29±4.10
vowel	79.54±4.01	82.41±3.72	86.48±3.25	90.04±3.02	86.59±3.19
waveform	76.36±1.77	78.78±1.77	82.19±1.59	84.47±1.54	83.71±1.62
yeast	59.46±3.40	59.76±3.46	59.46±3.41	59.50±3.42	60.14±3.69
zoo	92.61±7.33	93.10±7.48	96.07±5.89	92.85±7.07	93.41±7.28
mean value	83.26±3.85	84.35±3.84	84.66±3.74	84.85±3.60	84.93±3.76
overall rank	–	2.68	2.57	2.53	2.22
w/l/t	11-3-16	7-2-21	6-4-20	4-4-22	–

the training set, which is 25% of the full data set. A classifier is constructed from each local training set and bias, variance, and error are estimated on the test set. Since the bias-variance evaluation procedure utilizes training sets containing only 25% of each data set, we only do this decomposition on the 12 UCI data set which size is larger than 1000. Table 4 shows the detailed decomposition results of C4.5 , Naive Bayes and MACLEN using each of them as base learners respectively. Note that the actual bias-variance decomposition scheme of [16] generates a bias term that includes the intrinsic noise, so the errors shown in Table 4 are only composed of bias and variance.

Table 3. Accuracy and standard deviation of Naive Bayes, AdaBoost, AODE and MACLEN using Laplace estimation and M-estimation naive Bayes respectively, on the 30 UCI data sets. Bottom rows of the table present the mean values, overall ranks and Win-Loss-Tie (*w/l/t*) comparisons between MACLEN(L) and MACLAN(M) against other algorithms using pairwise *t*-tests at 95% significance level, respectively.

Data sets	Naive Bayes	AdaBoost	AODE	MACLEN(L)	MACLEN(M)
adult	84.07±0.42	85.19±0.41	85.20±0.41	85.04±0.39	85.10±0.37
anneal	96.13±2.16	99.32±0.86	98.05±1.37	96.85±1.93	97.11±1.79
autos	72.30±10.31	81.28±8.45	81.14±8.50	78.45±9.30	81.95±8.35
balance-scale	71.08±4.29	71.08±4.29	69.34±3.82	69.26±3.83	69.21±3.83
breast-cancer	72.94±7.71	68.87±8.64	72.73±7.01	72.35±7.10	70.95±7.69
car	85.46±2.56	90.25±2.48	91.41±2.06	91.51±2.06	91.76±2.04
colic	81.39±5.74	81.28±6.84	82.37±5.72	82.53±5.56	82.39±5.62
credit-a	86.25±4.01	85.83±3.92	86.55±3.82	86.32±3.82	86.23±3.89
credit-g	75.42±3.84	75.22±3.91	76.44±3.89	75.80±3.92	75.84±3.97
diabetes	77.85±4.67	77.84±4.74	78.07±4.56	78.06±4.51	78.02±4.50
glass	74.39±7.95	74.34±8.10	76.49±7.71	75.10±8.19	76.99±8.36
heart-c	83.60±6.42	83.26±6.48	83.26±6.19	83.33±6.25	83.33±6.29
heart-h	84.46±5.92	83.51±6.32	84.43±5.92	84.53±5.89	84.49±5.87
hepatitis	84.22±9.41	85.76±7.93	85.42±8.96	84.61±9.37	85.38±9.24
hypothyroid	98.48±0.59	99.32±0.41	98.74±0.54	98.55±0.56	98.78±0.54
ionosphere	90.77±4.76	93.62±4.08	92.97±4.32	92.99±4.07	93.32±3.83
iris	94.47±5.61	94.53±5.87	93.20±5.76	93.27±5.65	93.33±5.61
kr-vs-kp	87.79±1.91	95.14±1.23	91.03±1.66	89.24±1.86	89.29±1.87
letter	74.00±0.88	74.00±0.88	88.76±0.70	86.65±0.72	88.53±0.67
mushroom	95.52±0.78	100.0±0.00	99.95±0.07	99.35±0.29	99.81±0.16
nursery	90.30±0.72	91.91±0.76	92.73±0.62	92.60±0.62	92.60±0.63
primary-tumor	47.19±6.02	47.19±6.02	47.87±6.37	48.14±6.45	48.64±5.77
segment	91.71±1.68	93.21±1.46	95.77±1.24	95.06±1.40	96.48±1.14
sick	97.10±0.84	97.01±0.81	97.39±0.79	97.29±0.80	97.26±0.81
soybean	92.20±3.23	91.99±3.49	93.31±2.85	92.78±3.01	93.61±2.78
vehicle	62.51±3.81	62.52±3.81	72.31±3.62	71.44±3.46	72.24±3.38
vowel	65.23±4.53	72.08±4.73	80.88±3.81	79.96±4.09	83.72±3.73
waveform	80.72±1.50	80.72±1.50	86.03±1.56	82.25±1.31	82.34±1.30
yeast	59.16±3.80	59.16±3.80	59.72±3.86	59.54±3.85	59.81±3.79
zoo	93.21±7.35	93.61±7.02	94.66±6.38	94.66±6.38	97.21±5.13
mean value	81.66±4.11	82.97±3.97	84.54±3.80	83.92±3.89	84.52±3.76
overall rank	–	3.05	2.07	2.78	2.10
w/l/t	12-1-17	8-5-17	0-8-22	–	–
w/l/t	16-1-13	9-5-16	3-4-23	7-0-23	–

From Table 4, we can see that in most cases, the error reduction of MACLEN compared to its base learners is mainly due to its improvement of the bias. The mean values shown at the bottom of the table also clearly demonstrate this. In summary, the success of MACLEN mainly lies in that it can significantly reduce the bias of the base learner. This suggests that MACLEN is similar to Boosting style algorithm which has been shown that its improvement is mainly due to its ability to reduce the bias of the base learner.

Table 4. Bias-variance decomposition results of C4.5, Naive Bayes and MACLEN using each of them as base classifier respectively for the 12 UCI data sets which size are larger than 1000

Data sets	MACLEN(C4.5)			C4.5			MACLEN(NB)			Naive Bayes		
	Error	Bias	Var	Error	Bias	Var	Error	Bias	Var	Error	Bias	Var
adult	0.152	0.110	0.041	0.141	0.121	0.020	0.153	0.145	0.008	0.162	0.156	0.006
car	0.128	0.065	0.062	0.157	0.092	0.064	0.128	0.069	0.058	0.166	0.098	0.067
credit-g	0.284	0.182	0.101	0.290	0.204	0.085	0.245	0.192	0.052	0.246	0.195	0.050
hypothyroid	0.013	0.008	0.005	0.015	0.012	0.004	0.024	0.020	0.004	0.025	0.021	0.004
kr-vs-kp	0.015	0.008	0.007	0.020	0.011	0.009	0.121	0.094	0.026	0.136	0.109	0.027
letter	0.181	0.087	0.092	0.312	0.164	0.147	0.186	0.139	0.047	0.280	0.228	0.051
mushroom	0.000	0.000	0.000	0.000	0.000	0.000	0.012	0.010	0.002	0.062	0.058	0.003
nursery	0.049	0.021	0.028	0.066	0.038	0.028	0.075	0.063	0.011	0.094	0.083	0.011
segment	0.056	0.035	0.021	0.105	0.064	0.040	0.065	0.045	0.020	0.097	0.073	0.023
sick	0.025	0.019	0.006	0.025	0.019	0.006	0.024	0.021	0.002	0.025	0.022	0.002
waveform	0.237	0.122	0.115	0.281	0.159	0.120	0.179	0.166	0.013	0.191	0.179	0.012
yeast	0.431	0.317	0.113	0.453	0.328	0.124	0.419	0.338	0.080	0.419	0.339	0.079
mean	0.131	0.081	0.049	0.155	0.101	0.054	0.136	0.108	0.027	0.159	0.130	0.028

5 Conclusion

In this paper, we propose the MACLEN method, a new general ensemble method based on manipulating the class labels. It has several appealing properties: first, it can be applied to both binary and multi-class learning problems; second, its performance is superior to AdaBoost; third, it is simple, easy to parallelized and robust to noise. These demonstrate that manipulating class labels is also a general powerful way to generate strong ensemble besides the popular way of resampling the input instances or features.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. This work was supported in part by NKBRP(2005CB321905), NSFC (60873115) and AHEDUSF (2009SQRZ075).

References

1. Baxter, J.: A model for inductive bias learning. Journal of Artificial Intelligence Research 12(2), 149–198 (2000)
2. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases. University of California, Dept. of C.S., `http://www.ics.uci.edu.learn/MLRepository.html`
3. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
4. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
5. Caruana, R.: Multi-task learning. Machine Learning 28(1), 41–75 (1997)
6. Caruana, R., Sa, V.R.: Benefitting from the variables that variable selection discards. Journal of Machine Learning Research 3(2), 1245–1264 (2003)
7. Demsar, J.: Statistical Comparison of Classifiers over Multiple Data Sets. Journal of Machine Learning Research 7, 1–30 (2006)

8. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research 2, 263–286 (1995)
9. Dietterich, T.G.: Ensemble Methods in Machine Learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)
10. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Proceedings of the 13th International Conference on Machine Learning, pp. 123–140 (1996)
11. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning 29, 131–163 (1997)
12. Furnkranz, J.: Pairwise classification as an ensemble technique. In: Proceedings of the 13th European Conference on Machine Learning, pp. 97–110 (2002)
13. Goebel, M., Riddle, P.J., Barley, M.: A unified decomposition of ensemble loss for predicting ensemble performance. In: Proceedings of the 19th International Conference on Machine Learning, pp. 211–218 (2002)
14. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Trans. Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
15. Jiang, L., Zhang, H., Cai, Z.: A novel bayes model: hidden naive Bayes. IEEE Trans. Knowledge and Data Engineering 21(10), 1361–1371 (2009)
16. Kohavi, R., Wolpert, D.: Bias plus variance decomposition for zero-one loss functions. In: Proceedings of the 13th International Conf. on Machine Learning, pp. 275–283 (1996)
17. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. John Wiley and Sons, Chichester (2004)
18. Li, N., Yu, Y., Zhou, Z.H.: Semi-naive exploitation of one-dependence estimators. In: Proceedings of the 9th IEEE International Conf. on Data Mining, pp. 278–287 (2009)
19. Wang, Q., Zhang, L., Chi, M.M., Guo, J.K.: MTForest: Ensemble decision trees based on multi-task learning. In: Proceedings of the 18th European Conference on Artificial Intelligence, pp. 122–126 (2008)
20. Webb, G.I., Boughton, J., Wang, Z.: Not so naive bayes: Aggregating one-dependence estimators. Machine Learning 58(1), 5–24 (2005)
21. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco (2000)
22. Zhou, Z.H., Wu, J.X., Tang, W.: Ensembling neural networks: Many could be better than all. Artificial Intelligence 137(1), 239–263 (2002)

Supervised Learning with Minimal Effort

Eileen A. Ni and Charles X. Ling

Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7
{ani,cling}@csd.uwo.ca

Abstract. Traditional supervised learning learns from whatever training examples given to it. This is dramatically different from human learning; human learns simple examples before conquering hard ones to minimize his effort. Effort can equate to energy consumption, and it would be important for machine learning modules to use minimal energy in real-world deployments. In this paper, we propose a novel, simple and effective machine learning paradigm that explicitly exploits this important simple-to-complex (S2C) human learning strategy, and implement it based on C4.5 efficiently. Experiment results show that S2C has several distinctive advantages over the original C4.5. First of all, S2C does indeed take much less effort in learning the training examples than C4.5 which selects examples randomly. Second, with minimal effort, the learning process is much more stable. Finally, even though S2C only locally updates the model with minimal effort, we show that it is as accurate as the global learner C4.5. The applications of this simple-to-complex learning strategy in real-world learning tasks, especially cognitive learning tasks, will be fruitful.

Keywords: supervised learning, decision tree, minimal effort.

1 Introduction

Traditional supervised learning learns from whatever training examples given to it. This is dramatically different from human learning; human learns simple examples before conquering hard ones to minimize his effort. For example, infants learn from simple words and sentences first and gradually learn complex ones through repeated exposures from parents and care takers. Adults, on the other hand, often learn from simple to complex through repeated reviews of the same materials.

In the human learning research area, the "i+1" education theory suggests that human learns a small piece of new knowledge ("1") based on a large body of previously learned knowledge ("i") [1]. Originally, the "i+1" theory describes that the best methods to acquire a second language are those that supply "comprehensible input" and allow students to produce when they are ready [1,2]. We show that the "i+1" human learning theory can be applied in supervised machine learning: a learning algorithm will take less effort when it learns the next easiest thing based on the current learned knowledge.

In this paper, we propose a novel, simple and effective machine learning paradigm that explicitly exploits this important simple-to-complex learning strategy, called *S2C* (Simple to Complex). S2C builds its model with simple examples first. More specifically, it selects those examples that are close to the current model's prediction (thus simple), and updates the model with them. S2C works iteratively in this way which is almost the same as the human learning from simple to complex process.

Experiment results show that our new learning paradigm S2C has several distinctive advantages over C4.5 that takes the training examples in random order. First of all, S2C does indeed take much less effort in building its model than C4.5. This would be important in building machine learning modules that use minimal energy. Energy consumption control is highly important in real-world field deployments [3]. [4] indicates that when modules consume too much energy, they may run out of the batteries quickly and must be aggressively cooled; otherwise they would be unreliable and oftentimes be unavailable for use by the application scientists.

Second, minimal effort learning implies that the process of learning and the final learned model are more stable and reliable. This is certainly crucial for human learning, as well as for machine learning applications. To contrast the difference between S2C and C4.5 taking examples randomly, we also implement an "aggressive" learner who chooses examples from complex to simple. We show that the "aggressive" learner takes even more effort than C4.5.

Finally, even though S2C only locally updates the model with minimal effort, we show that it is as accurate as the global learner C4.5. One might think that as S2C always takes the simplest example to update its model locally and incrementally, it may not predict as accurately as the global learner C4.5 which builds its model on the whole dataset. Our experiment results show that S2C predicts only slightly worse than C4.5. In addition, we design a "mini-review" process, and add it into S2C. S2C then becomes less myopic and is shown to predict just as well as C4.5.

We explicitly exploit and implement this simple-to-complex learning strategy, a ubiquitous human learning strategy, in the machine learning research. Its applications in human-oriented learning tasks, especially cognitive learning tasks, will be fruitful.

The rest of this paper is organized as follows. Several previous works related to learning from simple to complex are reviewed in Section 2. Section 3 describes a generic S2C paradigm. We discuss an efficient implementation of S2C based on the decision tree learning algorithm in Section 4. Experiment results are shown in Section 5. We conclude our work in Section 6.

2 Related Work

Learning from simple to complex gradually may look similar to incremental learning previously studied [5,6], but they are much different. Incremental learning builds classifiers gradually based on whatever the data given passively. However, the S2C learning paradigm actively selects simpler examples first to learn, and then complex ones. However, the S2C learning paradigm is different from the traditional active learning [7,8]. Active learning selects the unlabeled examples that are most uncertain and acquires their labels from an oracle. Then it rebuilds a completely new model with all the labeled examples. The object of active learning is to learn a model with fewer labelings from the oracle. The S2C learning paradigm, on the other hand, builds a model with simple-to-complex labeled examples (no oracle is required) to reduce the effort.[1]

Ferr, etc. proposed delegating classifiers [9], and its main idea is divide-and-conquer. The learner builds the first classifier with all the training examples, and delegates those examples that cannot be predicted well (complex examples) to build delegating classifiers. However, it is difficult to determine how complex for examples to be delegated. Another potential issue is that the delegated examples probably are not sufficient which may affect the reliability of the delegating classifiers. However, S2C learns examples from simple to complex gradually. Thus S2C has neither threshold nor the difficulty of insufficient examples.

A similar idea to S2C is curriculum learning [10]. It learns by assigning the easier examples with higher weights first and then increasing the weights of the complex examples gradually, which is based on the idea that human or animal learns much better when the examples are organized in gradually more complex order [11]. The difficulty in this method is that the so-called easier or harder examples are given by human, which may be unreasonable and infeasible. However, S2C is a learner-centric paradigm. It means that it is the learner who decides what examples are simple or complex, which is much more realistic and appropriate.

Lifelong learning [12] addresses the situations in which a learner faces a series of different learning tasks providing the opportunity to transfer knowledge. Recently, a number of the transfer knowledge researches have been done in different applications, such as, Web document classification [13,14], sentiment classification [15], reinforcement learning [16], etc. The object of transfer learning is to transfer knowledge of other related, but different source data to the current learning data, such that a good model can be learned with fewer examples. The simple-to-complex strategy in S2C is similar to transfer learning. The difference is that it transfers knowledge that is learned from simple examples to complex examples, such that complex ones can be learned easier. From this perspective, S2C belongs to the vertical transfer learning more; the traditional transfer learning is more similar to the horizontal transfer learning in the psychology research area [17,18].

Deep structure learning attempts to learn high level features by the composition of lower level features [10]. It starts training on simpler human-crafted features and tries to get abstract high level features. One conceivable method is by training each layer one after the other [19,20]. It is also very different from S2C, as S2C learns simpler examples first and then complex examples.

[1] Reducing effort means less energy, e.g. power, that the algorithm needs to consume. IBM Research shows that power consumption is a major problem in designing computers to simulate human brain. `http://spectrum.ieee.org/computing/hardware/ibm-unveils-a-new-brain-simulator`

3 S2C Learning

Our novel simple-to-complex learning strategy models the adult iterative-learning process in which the same set of materials is repeatedly studied, and for each iteration only the currently simple cases are learned.

The rationale behind S2C is that learning is a gradual process of accumulating knowledge. Learning simple examples first may make the learning of harder examples easier, thus the whole learning process becomes easier (less effort) and more reliable. (See Section 3.2 for evaluation metrics for S2C).

3.1 S2C Learning Paradigm

At a high level, S2C is very simple, and it can use any classifier that can produce a refined class probability estimation (see later for details) as its base learner. Generally speaking, for each iteration, S2C selects the simplest example based on the current model and updates the model locally with the selected example, until all examples have been learned.

However, several important issues deserve further explanations. First, how can S2C select the simplest example? Second, how can S2C select the first simplest example before any model is built? Third, how can S2C select the simplest example when tie happens?

The first issue, how to select the simplest example, is crucial for the S2C learning paradigm. Without a proper measurement, S2C cannot choose the simplest example correctly. Here we propose a simple and effective measurement, which uses the prediction error of the current model for an example to measure how simple the example is. The smaller the error, the simpler the example is for the current model. Thus a base learner that can generate a refined class probability estimation is a requirement of S2C, as we mentioned before. Also the measurement is consistent with human intuition about simplicity: the less the surprise (i.e., difference or error), the simpler it is.

The second issue of selecting the first simplest example can be tricky, as the current model is empty. We design a simple and effective strategy for S2C. S2C scans over all the training examples to pick up the most frequent example. If no example appears more than once in the training set, then an example from the majority class will be chosen randomly.

The third issue, the tie-breaking strategy, can be crucial, if tie happens often when S2C selects the simplest example with the current model. If ties happen, S2C must choose one randomly. This may affect the performance of S2C. Indeed, for some algorithms, such as C4.5 (the base learner for S2C in this paper), ties do happen often. This is because C4.5 predicts all examples falling in the same leaf with the same prediction (i.e., same label and same probability estimation). For those algorithms, effective tie-breaking strategies are necessary. For C4.5, we use the Euclidean distance between positive and negative examples and the numbers of positive and negative examples in each node as heuristic information for the tie breaking. Details are presented in Section 4.

3.2 Measurements

In this subsection, the key issue we will discuss is how to evaluate S2C and compare it with other supervised learning algorithms. Since minimal effort is our target, effort is the crucial measurement in our paper. In addition, volatility is also important to measure how stable an algorithm is. Therefore, we present two measurements, effort and volatility, in the following.

Effort is a crucial measurement in this paper. *Effort* indicates how much work a learner has to do to learn certain examples. Intuitively the effort can be computed through energy usage, or computer time, etc. The problem is that it is difficult to measure them since they are related to hardware performances, the efficiency of programming language, etc. As a result, we choose two different methods to reflect the effort indirectly.

One method is using the prediction error to approximate the effort. We call it *error-based effort*, which is almost the same as the measurement for selecting the simplest example (in Section 3.1). The difference is that the measurement in Section 3.1 is used to measure how simple an example is, while the error-based effort here is about a learning model. At the same time, they are close related. If an example is the simplest one for a model, the model will take the least effort to learn it. The other method to reflect the effort is the size of the model being built, called *size-based effort.* Size can reflect how much effort is needed to build the model. Usually the larger the size of a model is, the more effort the model needs. For a decision tree, size-based effort can be the number of the nodes in the tree. These two (error-based effort and size-based effort) are good, universal approximation because they are independent of hardwares or other factors, and can be used to evaluate any learning algorithms easily.

The other important measurement, volatility, is used to evaluate how stable a learner is. If the error rate (accuracy) of a learner varies greatly from different runs, its volatility should be high. Volatility is thus the average value of the standard deviation of the prediction error rate for all of the current models. The volatility reflects the varying range of the error of a learner from different runs. Thus it is also one of the essential measurements to assess a learner.

We have presented the main strategies of the S2C learning paradigm in a high level. In the next section, we will present it with a specific base learner, C4.5.

4 S2C with Decision Tree

As we mentioned, the S2C learning paradigm can take most of the classification algorithms as its base learner easily. In this paper, we combine it with one of the most popular algorithms, C4.5. Originally, C4.5 builds a global tree in "one shot". However, S2C only provides C4.5 examples one by one in a simple-to-complex manner, such that it builds a tree locally and gradually. In the following, we will introduce this cooperation between S2C and C4.5 specifically.

As we discussed in Section 3.1, S2C selects one of the majority examples randomly (assume no repeated examples in datasets) for its base learner, C4.5, to build the first tree model - only a one-leaf tree. Then based on the current tree, S2C selects the simplest example, which is the one that is predicted most correctly by the current tree, and updates the current tree with it. S2C iterates this selecting and updating process until all the training examples are learned.

However, tie can happen often in decision trees, because a tree returns the label prediction and probability of an example according to the numbers of the positive and negative examples in the node that the example falls in. Thus, two different kinds of tie may happen when S2C selects the simplest example. One is that those examples that fall into the same leaf node. The other one is that those examples that fall into different nodes but have the same probability estimation. S2C could solve these tie problems by selecting one example randomly. However, inevitably this random manner would affect its performance severely. Better strategies are needed to judge which of the equally simple examples is simpler than the others. Two different tie-breaking strategies are given as follows.

Firstly, we design a novel tie-breaking strategy to solve the tie happening in the same leaf. In this situation, the equally simple examples belong to one class, say positive. The simplest positive example should be far away from any negative examples. Thus the tie-breaking strategy is to choose the example that has the furthest distance to its closest negative example comparing to other equally simple ones. That is to say, S2C calculates the Euclidean distance for each equally simple positive example to all the negative examples. It records the closest distance for each positive example, and selects the example having the greatest distance.

$$x_i = \arg\max_i(\min_j(d_{ij})) \qquad 0 \leq i < n_1; 0 \leq j < n_0. \tag{1}$$

where, d_{ij} is the distance from example x_i to x_j; x_i and x_j belongs to positive and negative respectively; n_1 is the number of the equally simple examples and n_0 is the number of the negative training examples.

In addition, an efficient strategy also is needed to break the ties among the examples that fall in different leaves but with the same probability estimation. Two factors can be considered for this tie-breaking. One is the number of the examples in the leaves that tie happens. Intuitively, the more examples a leaf has, the more reliable the leaf is in terms of probability estimation. The other factor is the different depths of leaves in the tree. Intuitively the closer a leaf is to the root, the more preferred the example that falls in the leaf is. For the first factor, if two positive examples fall into two positive leaves, say, A and B, and A has *9* positive examples and *1* negative, and B has *18* positive examples and 2 negative, the predictions for the two examples would be tied . However, B should be preferred over A when breaking this tie because B is more reliable. For the second factor, if a leaf C is on the first level and D is on the fifth level, and both C and D have, say, *9* positive examples and 1 negative, the example that falls in C should be preferred over the one that falls in D, since only one attribute is needed to predict its label. By combining the two factors, we propose Formula 2 as a heuristic for the tie-breaking. The preference of an example x_i can be defined as:

$$Preference = \frac{1 + \frac{N_1 - N_0}{N_{root}}}{2} * \frac{1}{L_{path}} \quad (2)$$

where, N_1 and N_0 are the numbers of the positive and negative examples respectively in the leaf node that x_i falls in; N_{root} is the number of the examples in the root node; L_{path} is the length from the root to the leaf that x_i falls in.

In Formula 2, the value of $\frac{1+\frac{N_1-N_0}{N_{root}}}{2}$ corresponds to the first factor we discussed. It indicates how positive x_i is. If $N_1 = N_0$, the value will be *1/2*; if N_1 is very small, its value will be close to zero (more negative); otherwise, it will be close to *1* (more positive). *1/Lpath* corresponds to the second factor. The closer a leaf to its root, the more preferred the example is.

During the iterative process of selecting the simplest example and updating the current tree, two issues need to be explained further. One is that if tie still happens with Formula 1 or Formula 2, S2C will select one example randomly. The other one is that S2C will only split the fringe node if needed when updating the current tree model. That is to say, if an example agrees with the fringe node it enters, S2C will not change the tree structure; otherwise it will split the fringe node according to the traditional C4.5 algorithm. This fringe-node-split strategy reduces the size-based effort effectively.

As we have discussed earlier, two measurements, effort (error-based and size-based effort) and volatility, can be used to evaluate the S2C paradigm effectively. Here we provide the details of how to evaluate the tree-based S2C in terms of effort first. The error-based effort can be calculated by the prediction error of the current model. For a decision tree, the prediction error of the current tree T_i for a positive example x_i is $(1 - p(1|x_i))$, where $p(1|x_i)$ is calculated by k/n. k is the number of the positive examples and n the total number of the examples in the node that x_i falls in. As discussed, another way to measure effort is the size-based effort. For a decision tree T_i, the size-based effort can be the number of the nodes in T_i. The more nodes in a tree, the more size-based effort is needed to build the tree.

The other important measurement, volatility, as discussed in Section 3.2, is the standard deviation of the prediction error rate . It indicates how volatile a learner is.

As we have presented, the S2C learning paradigm can easily take C4.5 as its base learner and works in a simple and effective manner. With the two new measurements, we will show that S2C indeed has several advantages over other learning algorithms in the next section.

5 Experiments

To illustrate the performance of S2C, we choose 10 UCI [21] datasets, including anneal, autos, breast_cancer, colic, diabetes, ecoli, glass, heart-h, sonar and vote, which are commonly used in the supervised learning research area. Originally autos and glass are multi-class. However, to compare with the other binary-class datasets, we transform them to binary-class datasets, autos_new and glass_new, by keeping the majority class and merging all the other classes. All the algorithms are implemented based on the WEKA [22] source code. In the experiments, 5-fold cross validation is used and the t-test results are with 95% confidence.

5.1 Comparison of Effort

We first compare S2C with C4.5 in terms of effort. C4.5 selects a new example from the training data randomly (not from simple to complex). In addition, instead of updating the current tree locally in S2C, C4.5 rebuilds a tree based on the current examples it has.[2]

To further contrast the differences between S2C and C4.5, we also implement a maximal effort learner, called *Jump-start*, which is very similar to S2C in framework but works in an opposite way. The Jump-start learner selects the most complex example (be predicted most wrongly) based on the current model and updates the current model with it iteratively until all the examples are learned. The rationale behind Jump-start is that the complex examples are more informative and useful to improve the current model. To make the comparison meaningful, Jump-start also takes C4.5 as its base learner.

As discussed in Section 4, we concern two kinds of effort, error-based effort and size-based effort. The error-based effort on the whole training set is the sum of the error when the learning algorithms process and learn every training example. Similarly, size-based effort of decision trees on the whole training set is the sum of the node number of the trees when each training example is learned.

We conduct the t-test on all the 10 datasets to compare the error-based effort between S2C, C4.5 and Jump-start. The results show that S2C takes much less error-based effort than C4.5 on 9 datasets, ties with C4.5 on only one dataset and loses on no dataset. However, the performance of Jump-start is poor. It loses to S2C and C4.5 on all of the 10 datasets. To further show the differences among the three algorithms, we present the average of error-based effort of the 5 runs on each dataset in the upper part of Table 1. On average C4.5 and Jump-start take about 1.3 times and 2 times as much as the error-based effort of S2C respectively. Thus it is clear that S2C needs much less error-based effort than C4.5 to learn all the training examples.

We also conduct the t-test on all the 10 datasets to compare the size-based effort. The results show that S2C wins C4.5 and Jump-start on all the 10 datasets significantly without exception, while Jump-start loses to S2C and C4.5 on all the datasets. We also present the average of size-based effort of the 5 runs on each dataset in the lower part of Table 1. On average, C4.5 and Jump-start take about 2.6 times and 6 times as much as the size-based effort of S2C. Obviously, S2C takes the least size-based effort to build a model, because it updates tree models locally instead of rebuilding trees, which saves much effort.

[2] We may also use ID5 [23], an incremental version of C4.5 to update the tree. However, it is shown [23] that ID5 produces identical tree as C4.5; thus we use C4.5 on the current training dataset directly.

Table 1. The comparison of effort among the three learning algorithms

	anneal	auto_n	b_cancer	colic	diabetes	ecoli	glass_n	heart-h	sonar	vote
					Error-based Effort					
S2C	6.9	7.6	10.0	16.0	41.0	7.0	11.6	13.3	12.5	7.1
C45	7.1	10.5	12.0	22.0	59.0	8.0	16.0	20.3	16.2	9.0
JS	23.7	14.5	23.0	30	73.5	19.3	20.1	22.3	21.4	15.9
					Size-based Effort					
S2C	3357	1198	3141	2841	21157	1478	1470	1792	2711	465
C45	14747	2937	12163	7482	49042	4399	4309	4665	3824	1034
JS	51327	7427	27501	14019	95623	12057	7343	9055	10107	3721

The experiment results have convinced us that the performances of S2C are excellent on both kinds of effort. Jump-start is the worst and C4.5 is in-between. Both the error-based effort and the size-based effort can be translated directly to how much energy that the computer running the program will take; however it is beyond this paper and will be our future work.

5.2 Comparison of Volatility

We have already learned that S2C needs much less effort to learn all the training examples. Less effort implies that the learned model should be less volatile. As discussed in Section 4, to compute the volatility, we run the each algorithm on each dataset for 5 times and compute the standard deviation of the error rate. We compare S2C with C4.5 and Jump-start and show the results in Table 2.

Table 2 shows the volatility of the three learning algorithms on each dataset. The volatility of S2C is almost 0 on all the datasets. On average, C4.5 is thousands of times more volatile than S2C, and Jump-start is about 3 times as volatile as C4.5. Thus it is clear that S2C is stable, while C4.5 and Jump-start are much more volatile. The reason is that no matter what sequence of the training examples is, S2C always results in a similar model. Learning algorithms with small volatility are important in learning real time data.

Also in Figure 1, we show that the error rate of S2C decreases more stably than C4.5 and Jump-start with more training examples taken in. We conduct the experiments on the 10 datasets, however, because of the limited space we only show the typical results of two datasets, colic and heart-h, in Figure 1. The x axis indicates the times of updating the trees and the y axis is about the error rate. Since the S2C model takes in all the examples of majority class first, it is only a one-leaf tree. The true building tree process only starts from taking in the first minority example. Thus its updating time is much less than that of C4.5 and Jump-start. From the figure, we can tell that the learning process of S2C is much shorter than the other two.

Figure 1 shows that with the increasing number of examples taken in, the error rate of Jump-start fluctuates very often at the beginning. C4.5 is a little

Table 2. The comparison of volatility among the three learning algorithms

	anneal	auto_n	b_cancer	colic	diabetes	ecoli	glass_n	heart-h	sonar	vote
S2C	5.6E-7	0.0	2.4E-8	3.1E-6	0.0	1.2E-8	0.0	4.7E-7	3.8E-6	5.6E-6
C45	0.011	0.032	0.016	0.003	0.024	0.039	0.039	0.032	0.067	0.020
JS	0.030	0.051	0.012	0.025	0.027	0.023	0.024	0.028	0.068	0.071

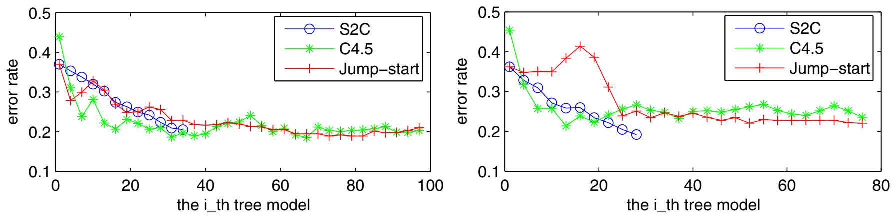


Fig. 1. The error rate on two datasets: (a) colic and (b) heart-h

better than Jump-start; however it still trembles at the beginning on most of the data. On the other hand, the error rate of S2C decreases more stably than the other two.

5.3 Comparison of Error Rate

As discussed in Section 4, S2C only updates the current tree at its fringe when the new simplest example is selected. It is extremely local and myopic. On the other hand, C4.5 rebuilds a tree with all the examples it has, and its final tree is built on the whole dataset. Thus C4.5 is a "global" tree. One might expect that S2C does not predict as well as C4.5. To compare the final error rate between S2C, C4.5 and Jump-start, we perform the t-test on the 10 datasets, after taking in the whole training set. We find that S2C is slightly worse than C4.5. It ties with C4.5 on 7 datasets, but loses on 3 datasets. The results were shown in the upper part of Table 3. This is expected as S2C updates the tree locally.

To avoid the greedy updating strategy of S2C without increasing the effort too much, we design a "mini-review" strategy. The strategy is that after learning K examples S2C will "review" them and use them together to expand the fringe of the tree. This "mini-review" process is similar to the summarizing process in human learning. The lower part of Table 3 gives the experiment results of the 10 datasets when $K=10$. It shows that S2C has almost the same low error rate as the "global" tree C4.5. Also we find that the error-rate performance of Jump-start is fairly good, and only loses to S2C and C4.5 on one dataset respectively. However, this similar performance costs Jump-start extraordinary effort.

The experiment results illustrate that S2C can work as well as the global algorithm C4.5 on error rate. However, it does take much less effort than C4.5 and Jump-start. In addition, S2C is much more reliable than the other two

Table 3. The t-test results on error rate (L: lose; T: tie; W: win)

	anneal	auto_n	b_cancer	colic	diabetes	ecoli	glass_n	heart-h	sonar	vote
					K = 1					
S2C vs. C45	L	L	T	T	T	L	T	T	T	T
C45 vs. JS	W	T	T	T	T	W	T	T	W	T
JS vs. S2C	T	T	T	T	T	L	T	T	T	T
					K = 10					
S2C vs. C45	T	T	T	T	T	T	T	T	T	T
C45 vs. JS	W	T	T	T	T	T	T	T	T	T
JS vs. S2C	L	T	T	T	T	T	T	T	T	T

algorithms. These are crucial to learning algorithms. All the experimental results show that the simple-to-complex learning strategy has distinctive advantages in the machine learning area.

6 Conclusions

In this paper we present the S2C learning paradigm, which exploits the simple-to-complex human learning strategy in the supervised learning research area, and implement it with C4.5 as its base learner. Experimental results show that S2C does take much less effort to learn a model than C4.5 and reaches very similar low error rate. Furthermore, S2C is much less volatile than C4.5. In our future work, we will apply S2C to more learning tasks, especially cognitive learning tasks.

References

1. Krashen, S.: The input hypothesis: Issues and implications. Addison-Wesley Longman Ltd., Amsterdam (1985)
2. Dong-lin, Z.: Krashen's Input Hypothesis and English classroom teaching
3. Mohiyuddin, M., Murphy, M., Oliker, L., Shalf, J., Wawrzynek, J., Williams, S.: A design methodology for domain-optimized power-efficient supercomputing. In: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, pp. 1–12. ACM, New York (2009)
4. Hsu, C., Feng, W., Archuleta, J.: Towards efficient supercomputing: A quest for the right metric. In: Proceedings of the High-Performance Power-Aware Computing Workshop, Citeseer (2005)
5. Lange, S., Grieser, G.: On the power of incremental learning. Theoretical Computer Science 288(2), 277–307 (2002)
6. Giraud-Carrier, C.: A note on the utility of incremental learning. AI Communications 13(4), 215–223 (2000)
7. Baram, Y., El-Yaniv, R., Luz, K.: Online choice of active learning algorithms. The Journal of Machine Learning Research 5, 255–291 (2004)
8. Cohn, D., Ghahramani, Z., Jordan, M.: Active learning with statistical models, Arxiv preprint cs/9603104 (1996)

9. Ferri, C., Flach, P., Hernández-Orallo, J.: Delegating classifiers. In: Proceedings of the twenty-first international conference on Machine learning. ACM, New York (2004)
10. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning. ACM, New York (2009)
11. Krueger, K., Dayan, P.: Flexible shaping: how learning in small steps helps. Cognition 110(3), 380–394 (2009)
12. Thrun, S.: Is learning the n-th thing any easier than learning the first? Advances in Neural Information Processing Systems, 640–646 (1996)
13. Fung, G., Yu, J., Lu, H., Yu, P.: Text classification without negative examples revisit. IEEE Transactions on Knowledge and Data Engineering 18(1), 6–20 (2006)
14. Sarinnapakorn, K., Kubat, M.: Combining Subclassifiers in Text Categorization: A DST-Based Solution and a Case Study. IEEE Transactions on Knowledge and Data Engineering 19(12), 1638–1651 (2007)
15. Blitzer, J., Dredze, M., Pereira, F.: Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: Annual Meeting-Association For Computational Linguistics, vol. 45, p. 440 (2007)
16. Taylor, M., Stone, P.: Cross-domain transfer for reinforcement learning. In: Proceedings of the 24th international conference on Machine learning, p. 886. ACM, New York (2007)
17. Rebello, N., Cui, L., Bennett, A., Zollman, D., Ozimek, D.: Transfer of learning in problem solving in the context of mathematics and physics. In: Learning to Solve Complex Scientific Problems. Lawrence Earlbaum, Mahwah (2007)
18. Gagné, R.: The conditions of learning. Holt, Rinehart & Winston, New York (1970)
19. MarcAurelio Ranzato, Y., Boureau, L., LeCun, Y.: Sparse feature learning for deep belief networks. Advances in neural information processing systems 20
20. Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring strategies for training deep neural networks. The Journal of Machine Learning Research 10, 1–40 (2009)
21. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
22. WEKA Machine Learning Project: Weka, `http://www.cs.waikato.ac.nz/~ml/weka`
23. Utgoff, P.: Improved training via incremental learning. In: Proceedings of the sixth international workshop on Machine learning table of contents, pp. 362–365. Morgan Kaufmann Publishers Inc., San Francisco (1989)

Generating Diverse Ensembles to Counter the Problem of Class Imbalance

T. Ryan Hoens and Nitesh V. Chawla

The University of Notre Dame, Notre Dame, IN 46556, USA
{thoens,nchawla}@nd.edu

Abstract. One of the more challenging problems faced by the data mining community is that of imbalanced datasets. In imbalanced datasets one class (sometimes severely) outnumbers the other class, causing correct, and useful predictions to be difficult to achieve. In order to combat this, many techniques have been proposed, especially centered around sampling methods. In this paper we propose an ensemble framework that combines random subspaces with sampling to overcome the class imbalance problem. We then experimentally verify this technique on a wide variety of datasets. We conclude by analyzing the performance of the ensembles, and showing that, overall, our technique provides a significant improvement.

1 Introduction

A common problem faced in data mining is dealing with "imbalanced datasets", or datasets in which one class vastly outnumbers the other in the training data. For most classifiers this causes a problem, as merely classifying every instance as the majority class present in the training data can result in very high accuracy. Therefore when dealing with imbalanced datasets, sampling methods such as oversampling, undersampling, and sampling by synthetically generating instances, SMOTE, have become standard approaches for improving classification performance [1]. Performance is improved as the sampling methods alter the training data distribution, biasing the data towards the minority class.

In addition to sampling methods, ensemble methods [2] have also been used to improve performance on imbalanced datasets. They combine the power of multiple (usually weak) classifiers trained on similar datasets to provide accurate predictions for future instances. The training data is often varied in such a way as to give each classifier a (slightly) different dataset so as to avoid overfitting. The popular ensemble methods, bagging [3] and boosting [4], have been adapted with sampling strategies to counter the issue of high class imbalance [5–7]. These methods work on the entire feature space as they focus on modifying the selection of instances in the ensemble training sets.

We posit that sampling methods, especially SMOTE, might stand to gain more when working in a reduced feature space as doing so will not only inject more diversity into the ensemble via the learning algorithm, but also via the bias of the sampling algorithm. To this end, we propose an extension to the random subpace method [8] that integrates SMOTE (in Section 2). We test on 21 widely available datasets with varying degrees of class imbalance (Section 4 includes the results). We comprehensively compare the proposed random subspace and SMOTE combination with bagging, AdaBoost, random subspaces, random forest, and the combination of random subspace and undersampling. Using the statistical tests recommended by Demšar, we determine the statistical significance of results. We show that sampling methods, combined with random subspaces are more effective than the other combinations of ensemble and sampling methods. We explain this behavior by invoking the notion of diversity.

To summarize, **the contributions** of this paper are as follows:

1. An extension to the random subspace method to deal with the class imbalance problem.
2. Empirical evaluation on a wide variety of imbalanced datasets, and establishing the superiority of the new ensemble framework.
3. Analyzing the performance of the methods using the measures of diversity.

2 Using Random Subspaces to Improve Performance on Imbalanced Datasets

The random subspace method (RSM) [8], described in Algorithm 1, is an ensemble method in which multiple classifiers are learned on the same dataset, but each classifier only actively uses a subset of the available features.

Because each classifier learns on incomplete data, each individual classifier is less effective than a single classifier trained on all of the data. However since the RSM combines multiple classifiers of this type, each with its own bias based on the features it sees, it sees an increase in performance over the base classifier.

Algorithm 1. The random subspace method.

```
Require: Training set $X$ with $n$ instances and $m$ features, number of features to consider $0 < m' < m$, and number of classifiers to train $e > 0$.
Ensure: CLASSIFIER is the model trained on training set $X$, consisting of $e$ classifiers.
  for $i = 1$ to $e$ do
    Select $F$, a random subset of the features such that $|F| = m'$.
    Let $X' \leftarrow X$.
    for all $f$ such that $f$ is a feature of $X'$ do
      if $f \notin F$ then
        Remove feature $f$ from $X'$
      end if
    end for
    Train CLASSIFIER$_i$ on dataset $X'$.
  end for
```

2.1 Random Subspace Method + Sampling

Sampling is a popular methodology to counter the problem of class imbalance. However sampling methods, especially SMOTE, work with the entire set of features or dimensions and may not be as effective in reducing the sparsity of the data. High

Algorithm 2. The random subspace method with an added resampling step.

```
Require: Training set $X$ with $n$ instances and $m$ features, number of features to consider $0 < m' < m$, number of classifiers to train $e > 0$, and sampling function $S$ which takes a dataset as a parameter, and returns a dataset.
Ensure: CLASSIFIER is the model trained on training set $X$, consisting of $e$ classifiers.
  for $i = 1$ to $e$ do
    Select $F$, a random subset of the features such that $|F| = m'$.
    Let $X' \leftarrow X$.
    for all $f$ such that $f$ is a feature of $X'$ do
      if $f \notin F$ then
        Remove feature $f$ from $X'$
      end if
    end for
    $X'_s \leftarrow S(X')$.
    Train CLASSIFIER$_i$ on dataset $X'_s$.
  end for
```

dimensionality is an Achilles' heel for sampling approaches like SMOTE, as they can lead to a higher degree of variance given low density and separation among classes because of the features' spread. Intuitively, applying SMOTE to reduced subspaces at a time will also control the amount of variance that SMOTE may introduce.

Sampling methods consider the class skew and properties of the dataset as a whole. Datasets often exhibit characteristics and properties at a local, rather than global level. Hence, it becomes important to analyze and consider the datasets in the reduced subspace. We also posit that by first randomly selecting a reduced subspace, sampling along that subspace, and then learning a classifier will induce a higher diversity, which is a necessary condition for improved performance of classifiers.

To this end we propose using SMOTE within each randomly selected subspace. Since SMOTE creates synthetic instances by interpolating feature values based on neighbors, we see that it is dependent upon the distance metric used to determine nearest neighbors. Thus by removing features from the feature space, we see that we are altering the distance between instances, and therefore (potentially) changing the way in which each classifier modifies its training data. That is, we create a hybrid of the RSM by combining it with SMOTE to create RSM+SMOTE. Just like in the RSM during the training phase each classifier is trained on a subset of the data in which some features are removed. After removing a subset of the features, SMOTE is then applied to the dataset which is subsequently used to train the classifier. Since SMOTE is dependent upon the features, and since in ensemble methods having classifiers with different biases is optimal, this should provide better performance over other techniques.

A generalized version of this algorithm (RSM+sampling) is given in Algorithm 2. In this algorithm as opposed to explicitly using SMOTE, a generic sampling method is supplied as a parameter for use in the algorithm. That is, RSM+SMOTE is a special case of RSM+sampling curried with SMOTE as the sampling method. In our tests we also consider using random undersampling as the sampling method. This lacks the appeal of SMOTE, however, as it does not depend upon the subspace chosen, and is thus more like random forests.

Note that while in principal any classifier can be learned, we use C4.5 decision trees as is common in the literature. In order to determine the predicted class of the ensemble, we consider simple majority voting of the classifiers, similar to the other ensemble methods discussed. As an area of future work Chawla and Sylvester [9] outline a procedure for weighted voting as applied to imbalance datasets. We will also be including Hellinger Distance Decision Trees as part of future work [10].

Complexity Analysis. When discussing ensemble methods, an important consideration is the method's computational complexity. In this section we therefore give an overview of the complexity of the scheme considered in the paper. That is we only consider the case of RSM+SMOTE with C4.5 decision trees as the classifier.

Let us now consider building an ensemble of e C4.5 decision trees using random subspaces and SMOTE on a dataset D consisting of n instances and m features. From the pseudo-code given in Algorithm 2, we see that each iteration of the for-loop first selects m' features $O(m')$. The algorithm then (in our case) applies SMOTE which has a complexity of $O(n_p^2)$, where n_p denotes the number of positive (minority) class examples in D. Applying SMOTE to the dataset results in additional instances being added to the dataset, leaving us with n' instances. Finally a C4.5 decision tree is learned on the resulting dataset which has complexity $O(n'm' \log n')$. Combining this, we see that each iteration of the loop has a complexity of $O(m' + n_p^2 + n'm' \log n') = O(n_p^2 + n'm' \log n')$. Therefore the complexity of constructing RSM+SMOTE is $O(e \cdot (n_p^2 + n'm' \log n'))$.

As with most ensembles, we can achieve an order e speedup to our method if we instead build each tree in parallel. That is we break the task up into e jobs and distribute the tree building process to e cores. Additionally since the voting function is simple majority voting, the trees can remain distributed if the cost of transferring them is too high.

Table 1. Legend of abbreviations

Full Name	Abbreviation
Bagging	BG
AdaBoost	BT
Random Subspace Method	RSM
Random Forest	RF
Synthetic Minority Over-sampling Technique	SMOTE
Undersampling	usamp
Random Subspace Method with SMOTE	RSM+SMOTE
Random Subspace Method with Undersampling	RSM+usamp

3 Experimental Design

We used the open source tool Weka, and implemented our classifier RSM+sampling. In order to test the robustness of our method compared to existing methods, we

Table 2. Details of the datasets in this report

Dataset	# Features	# Examples	CV	Dataset	# Features	# Examples	CV
boundary	175	3505	0.93	hypo	25	3163	0.90
breast-w	9	699	0.31	ism	6	11180	0.95
breast-y	9	286	0.41	oil	49	937	0.91
cam	132	18916	0.90	page	10	5473	0.80
compustat	20	13657	0.92	phoneme	5	5404	0.41
covtype	10	38500	0.86	PhosS	480	11411	0.89
credit-g	20	1000	0.40	pima	8	768	0.30
estate	12	5322	0.76	satimage	36	6430	0.81
fourclass	2	862	0.29	SVMguide1	4	3089	0.29
germannumer	24	1000	0.40	tic-tac-toe	9	958	0.31
heart-v	13	200	0.49				

included AdaBoostM1 (BT), bagging (BG), Random Forests (RF), and Random Subspaces (RSM) in our experiments. We also applied SMOTE and undersampling (usamp) to the entire dataset and then learned a single classifier. For all of the methods we use the J48 decision tree as the base classifier with Laplace smoothing and no pruning. As each of the ensemble methods requires a number of classifiers to train, we train 100 classifiers for each. See Table 1 for the entire set of classifiers used in this paper.

We evaluated each of the classifiers on the twenty-one datasets from a number of different resources, including finance, biology, oil spill, medicine, UCI and LibSVM data repositories (Table 2) [10–12]. Similar to [13], we use the coefficient of variation ("CV") to measure imbalance. To determine the CV, we calculate the ratio of the mean and standard deviation of the class counts in the dataset. As we seek to determine our classifiers' performance on imbalanced datasets, we choose to test on datasets with a CV above 0.28. This corresponds to a dataset for which less than 35% of the instances are minority class. In general, the larger the value of CV, the more imbalanced the dataset.

3.1 Experiments

In order to compare the classifiers, we use 10-fold cross validation. In 10-fold cross validation, each dataset is broken into 10 disjoint sets such that each set has (roughly) the same distribution. The classifier is learned 10 times such that in each iteration a different set is withheld from the training phase, and used instead to test the classifier. We then compute the AUROC (Area Under the Receiver Operating Characteristic) value as the average of each of these runs.

3.2 Statistical Tests

While many different techniques have been applied to attempt to compare classifier performance across multiple datasets, Demšar suggests comparisons based on ranks. Following the strategy outlined in [14], we rank the performance of each classifier by its average AUROC, with 1 being the best. Since we seek to determine whether or not our new methods are statistically significantly better than the existing methods, we use the Friedman and Bonferroni-Dunn tests [14].

The Friedman test is first applied to determine if there is a statistically significant difference between the rankings of the classifiers. That is, it tests to see if the rankings are not merely randomly distributed. Next, as recommended by Demšar, we preform the Bonferroni-Dunn test to compare each classifier against the control classifier.

4 Results

From Table 3 it is apparent that RSM+SMOTE performs significantly better than the other classifiers, achieving an average ranking that is over a full rank better than the next best classifier, RSM+usamp. This is further reinforced by noting that RSM+SMOTE is the best classifier in 12 of the 21 datasets, and second best on 5. RSM+SMOTE not only outperforms bagging and boosting, but also applying SMOTE on the entire dataset before learning a classifier. The results of using SMOTE on the entire dataset are not included in this paper due to space restrictions, as well as to keep the focus on ensemble based methods.

Out of all of the datasets, RSM+SMOTE achieves its worst rank of 4 on the fourclass dataset. This is unsurprising, however, as fourclass is not a suitable dataset to use RSM+SMOTE on, as it only has 2 features. Since RSM chooses a subspace of the features to learn on, it is necessary that there are enough features to gain power from only

Table 3. Rank of each classifier on the datasets

	RSM+SMOTE	RSM+usamp	RF	RSM	BG	BT
breast-w	2.0	1.0	3.0	4.0	5.0	6.0
breast-y	3.0	1.0	5.0	2.0	4.0	6.0
pima	1.0	2.0	5.0	3.0	4.0	6.0
hypo	1.0	2.0	5.0	3.0	4.0	6.0
phoneme	3.0	5.0	1.0	4.0	2.0	6.0
ism	1.0	2.0	4.0	3.0	5.0	6.0
SVMguide1	2.0	4.0	5.0	3.0	1.0	6.0
cam	1.0	2.0	5.0	3.0	4.0	6.0
credit-g	1.0	3.0	4.0	2.0	5.0	6.0
ion	1.0	3.0	4.0	2.0	5.0	6.0
covtype	1.0	5.0	2.0	4.0	3.0	6.0
satimage	2.0	3.0	1.0	4.0	5.0	6.0
PhosS	2.0	1.0	5.0	3.0	4.0	6.0
fourclass	4.0	5.0	1.0	6.0	3.0	2.0
tic-tac-toe	3.0	5.0	1.0	6.0	2.0	4.0
compustat	1.0	4.0	2.0	5.0	3.0	6.0
estate	1.0	3.0	4.0	5.0	2.0	6.0
heart-v	2.0	1.0	5.0	3.0	4.0	6.0
boundary	1.0	2.0	4.0	3.0	5.0	6.0
oil	1.0	3.0	4.0	2.0	5.0	6.0
page	1.0	2.5	5.0	2.5	4.0	6.0
Average	1.667	2.833	3.571	3.452	3.762	5.714

Table 4. Table denoting whether or not RSM+SMOTE is statistically significantly better than all other classifiers at various confidence levels on the datasets

Confidence Level	RSM+usamp	RF	RSM	BG	BT
99%		✓	✓	✓	✓
95%	✓	✓	✓	✓	✓
90%	✓	✓	✓	✓	✓

considering such a subset. Since fourclass only has 2 features, however, each classifier only classifies based on one feature. Similarly when SMOTE is applied to the feature there is only one axis of freedom, and the effect is that of placing instances somewhat randomly on a line. Intuitively this does not seem like an optimal strategy, and explains the poor performance of the classifier on the dataset. This also leads us to not recommend using RSM+SMOTE on such low dimensional datasets.

RSM+usamp also performs very well when compared to the other classifiers. On 10 of the 21 datasets it obtains the best or second best AUROC among the tested methods. This points to the sampling of individual subspaces to be a robust technique, as the two presented techniques offer a great advantage over the other classifiers.

In order to measure the statistical significance of the results, we use the methods outlined in Section 3.2. The results of the tests are presented in Table 4. In this figure we show the results of the Friedman and Bonferroni-Dunn tests which show that RSM+SMOTE performs statistically significantly better than all classifiers at the 95% confidence level. At 99% confidence it outperforms all of the classifiers except for RSM+ usamp. This is a strong result as it shows that the RSM can benefit greatly from applying sampling at the individual classifier level in addition to using a subspace of the features.

5 Analysis of Ensembles Using Diversity

When considering an ensemble of classifiers, the ideal situation would be a case where no two classifiers agree on instances on which they err. Given the nature of classifiers and data, however, achieving this ideal is highly unlikely. In order to determine how close an ensemble comes to the ideal, we measure the diversity by the degree to which an ensemble of classifiers disagree on such instances. Kuncheva and Whitaker [15] show that such diversity is an important property for achieving a good performance from ensembles.

In order to measure the diversity of the ensembles, we use the κ metric defined by Dietterich [16] as:

$$\begin{aligned}
\Theta_1 &= \frac{\sum_{i=1}^{T} C_{ii}}{m}, \\
\Theta_2 &= \sum_{i=1}^{T} \left(\sum_{j=1}^{T} \frac{C_{ij}}{m} \sum_{j=1}^{T} \frac{C_{ji}}{m} \right), \\
\kappa &= \frac{\Theta_1 - \Theta_2}{1 - \Theta_2},
\end{aligned}$$

where T is the number of classes, C is a $T \times T$ matrix such that C_{ij} denotes the number of instances assigned to class i by the first classifier and class j by the second classifier, and m is the number of instances. Given this, Θ_1 measures the degree of agreement, and Θ_2 is the degree of agreement expected at random. κ is then the measure of diversity, where 0 denotes agreement purely by chance, and 1 (-1) denotes perfect agreement (disagreement). The κ values can then be plotted against the accuracy for each pair of classifiers in the ensemble in order to obtain a graphical representation of the diversity of the ensemble. The x-axis is the κ value and the y-axis is the accuracy.

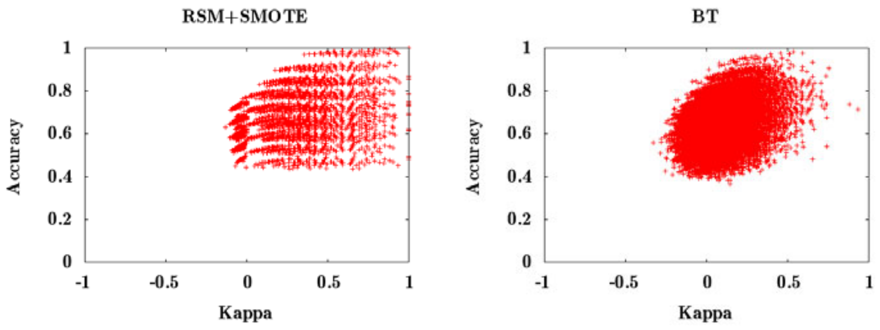


Fig. 1. κ plots for RSM+SMOTE (left) and AdaBoost (right) on the oil dataset

We now show some of the representative results explaining the behavior. We consider three representative results. Figure 1 shows the κ plot of RSM+SMOTE and AdaBoost on the oil dataset. Given the two plots, it becomes apparent why RSM+SMOTE outperforms AdaBoost on this dataset. While RSM+SMOTE shows some agreement among the classifiers along with high accuracy, AdaBoost shows much lower agreement (in fact almost completely random agreement) with similar accuracy. Since the classifiers in AdaBoost are agreeing almost at random, and the problem is very imbalanced, this is close to what we would expect of random classifiers which are biased towards the majority class. Alternatively, RSM+SMOTE shows very high accuracy with non negligible agreement. This points to highly accurate classifiers which agree when they are correct, and disagree on incorrectly classified instances, as desired.

While the previous example showed RSM+SMOTE performing well, Figure 2 depicts it being outperformed by RF. While AdaBoost underperformed for being too dissimilar, RSM+SMOTE underperformed for being too similar. As can be seen in the figure, RSM+SMOTE has a cluster of points at $\kappa = 1$. This means that the classifiers were in perfect agreement on every example. This adversely affects the diversity of the ensemble, effectively adding weighted classifiers to the ensemble. RF, on the other hand, shows highly accurate, yet diverse, classifiers.

Finally we show an instance where RSM+SMOTE outperforms RSM+usamp in Figure 3. This plot shows precisely the desired results for RSM+SMOTE. That is, since the classifiers are highly accurate, they should have a high κ value which is approximately twice the error rate. While the κ values are in the higher strata, none of them are actually 1, which means that classifiers never have a perfect agreement and are diverse, despite high accuracies. RSM+usamp, on the other hand, shows similar high

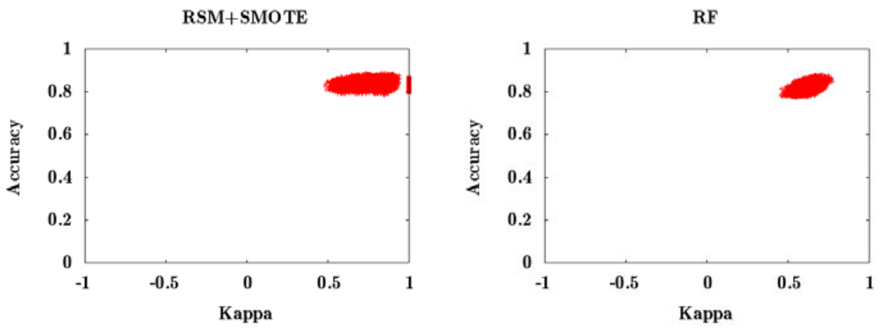


Fig. 2. κ plots for RSM+SMOTE (left) and RF (right) on the phoneme dataset

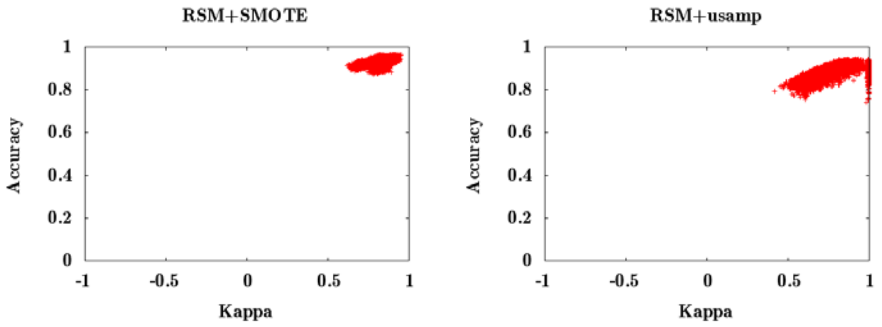


Fig. 3. κ plots for RSM+SMOTE (left) and RSM+usamp (right) on the covtype dataset

accuracies, yet many classifiers overlap as demonstrated by the number of points at $\kappa = 1$. As aforementioned this directly affects the diversity, and it is therefore unsurprising that RSM+SMOTE outperformed RSM+usamp.

6 Related Work

One popular approach towards improving performance on classification problems is to use ensembles. When using ensembles one attempts to leverage the classification power of multiple classifiers (learned on different subsets of the training data), to overcome the downsides of traditional classification algorithms, such as over fitting. Dietterich [2] provides a broad overview as to why ensemble methods often outperform a single classifier. In fact Hansen and Salamon [17] prove that under certain constraints (the average error rate is less than 50% and each classifier is erroneous independent of the others), the expected error rate of an instance can go to zero as the number of classifiers goes to infinity. Thus when seeking to build multiple classifiers, it is optimal to ensure that the classifiers are diverse. One way of ensuring this is by modifying the training data each classifier is learned on.

Two techniques for modifying the training data used by each classifier are bagging [3] and AdaBoost [4]. In bagging, introduced by Breiman, each training set is chosen by sampling (with replacement) from the original training set T. In AdaBoost introduced by Freund and Schapire, however, each training set is iteratively chosen based on the difficult to classify instances. That is classifiers are iteratively learned, and misclassified instances are more likely to be chosen in later training sets.

Both of the aforementioned techniques have their advantages and disadvantages. Bagging is trivially parallelizeable, and thus more amenable to building a large ensemble, however does not reduce the bias. AdaBoost, on the other hand reduces both variance and bias and has theoretical guarantees, but is sensitive to noise. It is therefore dependent upon the dataset which technique will provide better results. Because of this, Kotsiantis and Pintelas [18] devise a methodology of combining both bagging and AdaBoost in order to create a better ensemble of classifiers.

While the previous techniques modified the training set by altering the distribution of examples, the random subspace method [8] modifies the examples themselves. That is, when attempting to build an ensemble with n classifiers, each classifier independently chooses a subset of the features to use for training. Thus while most classifiers suffer from the curse of dimensionality, the random subspace method mitigates this by pruning the feature space.

Finally the random forest technique [19] combines the above techniques by using bagging to create the training sets, and then (optionally) applying random subspaces to each training set individually before learning the final classifier. This has the effect of generally producing a highly effective ensemble on most datasets, as combining multiple different sources of randomness leads increased diversity of the training data, and thus increased diversity of the ensemble.

In addition to applying these traditional ensemble methods to the class imbalance problem, modified versions have also been developed. Chawla et al. introduce SMOTEBoost [5], which combines boosting with SMOTE by, during each iteration of boosting, using SMOTE on the hard to classify instances. Guo and Viktor develop another extension for boosting, DataBoost-IM, which identifies hard instances in order to generate similar synthetic examples, and then reweights the instances to prevent a bias towards the majority class [7]. Liu, Wu, and Zhou propose two methods, EasyEnsemble and BalanceCascade [6], which generate training sets by choosing an equal number of majority and minority class instances from a larger training set. Finally Hido and Kashima introduce a variant of bagging, “Roughly Balanced Bagging” (RB bagging) [20] which alters bagging to emphasize the minority class.

In addition to ensemble methods to deal with the class imbalance problem, sampling techniques can also be employed. The two simplest sampling techniques are oversampling and undersampling. In oversampling, the minority class instances are replicated to create a larger minority class set. Conversely, in undersampling majority class instances are removed in order to level the class distribution. Both of these techniques, among others, have been widely employed [21–23]

A more sophisticated technique which seeks to combat class imbalance is known as SMOTE (Synthetic Minority Over-sampling Technique) [24]. When applying SMOTE, the training set is altered by adding more minority class examples in order to force the class distribution to become more balanced. Instead of simply oversampling the minority class (i.e., duplicating minority examples at random), SMOTE creates new synthetic minority examples by first selecting a minority example and its k nearest neighbors. The synthetic example is then created by choosing a neighbor and an feature. As the two examples form a line segment in the chosen feature space, the synthetic example's value for the feature is chosen to lie along that line segment.

7 Conclusion

We proposed an extension to the RSM to mitigate the effects of class imbalance, called RSM+Sampling, proposing SMOTE be used as the default sampling method. We then compared this method and RSM+usamp against bagging, AdaBoost, random forest, and the random subspace method. We posited that by applying SMOTE to subspaces and then learning classifiers will lead to an improved performance due to more diverse classifiers as well as less noise imputation due to SMOTE. The latter arises as SMOTE is only applied to a much reduced set of features at a time, and is thus a more controlled phenomenon. It is also not affected by the data sparsity and high dimensionality. To test this hypothesis we ran experiments on 21 widely available imbalanced datasets.

The results on the selected datasets showed RSM+SMOTE outperformed all other classifiers tested at the 95%, confidence level, and only did not outperform RSM+usamp at the 99% confidence levels. From these results it is apparent that RSM+SMOTE is well suited to overcoming the class imbalance problem. We argue that this is due to the diversity that SMOTE adds to the ensemble. That is SMOTE adds perturbations to the training data based on the features which has a positive effect on the diversity of the ensemble, and therefore increases performance. Based on this and the statistical significance tests, we recommend the use of RSM+SMOTE on imbalanced datasets.

Acknowledgements

Work was supported in part by the NSF Grant ECCS-0926170 and the Notebaert Premier Fellowship.

References

1. Chawla, N.V., Cieslak, D.A., Hall, L.O., Joshi, A.: Automatically countering imbalance and its empirical relationship to cost. Data Mining and Knowledge Discovery 17(2), 225–252 (2008)
2. Dietterich, T.G.: Ensemble methods in machine learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)
3. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
4. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: Thirteenth International Conference on Machine Learning (1996)
5. Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: Smoteboost: improving prediction of the minority class in boosting. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 107–119. Springer, Heidelberg (2003)
6. Liu, X.Y., Wu, J., Zhou, Z.H.: Exploratory under-sampling for class-imbalance learning. In: ICDM '06: Proceedings of the Sixth International Conference on Data Mining, Washington, DC, USA, pp. 965–969. IEEE Computer Society, Los Alamitos (2006)

7. Guo, H., Viktor, H.L.: Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. In: SIGKDD Explorations, pp. 30–39. ACM, New York (2004)
8. Ho, T.: The random subspace method for constructing decision forests. Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
9. Chawla, N.V., Sylvester, J.: Exploiting diversity in ensembles: Improving the performance on unbalanced datasets. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 397–406. Springer, Heidelberg (2007)
10. Cieslak, D.A., Chawla, N.V.: Learning decision trees for unbalanced data. In: Daelemans, W., et al. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 241–256. Springer, Heidelberg (2008)
11. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
12. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
13. DeGroot, M., Schervish, M.: Probability and Statistics, 3rd edn. Addison-Wesley, Reading (2001)
14. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
15. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles. Machine Learning 51, 181–207 (2003)
16. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning 40(19), 139–157 (2000)
17. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence 12(10), 993–1001 (1990)
18. Kotsiantis, S., Pintelas, P.: Combining bagging and boosting. International Journal of Computational Intelligence 1(4), 324–333 (2004)
19. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
20. Hido, S., Kashima, H.: Roughly balanced bagging for imbalanced data. In: Statistical Analysis and Data Mining, pp. 143–152. SIAM, Philadelphia (2008)
21. Drummond, C., Holte, R.C.: C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. In: ICML Workshop on Learning from Imbalanced Datasets II (2003)
22. Weiss, G.M., Provost, F.: Learning when training data are costly: the effect of class distribution on tree induction. Journal of Artifical Intelligent Research 19, 315–354 (2003)
23. Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explorations 6, 20–29 (2004)
24. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research 16, 321–357 (2002)

Relationship between Diversity and Correlation in Multi-Classifier Systems

Kuo-Wei Hsu and Jaideep Srivastava

University of Minnesota, Minneapolis, MN 55455, USA
{kuowei,srivasta}@cs.umn.edu

Abstract. Diversity plays an important role in the design of Multi-Classifier Systems, but its relationship to classification accuracy is still unclear from a theoretical perspective. As a step towards the solution of this probelm, we take a different route and explore the relationship between diversity and correlation. In this paper we provide a theoretical analysis and present a nonlinear function that relates diversity to correlation, which hence can be further related to accuracy. This paper contributes to connecting existing research in diversity and correlation, and also providing a proxy to the relationship between diversity and accuracy. Our experimental results reveal deeper insights into the role of diversity in Multi-Classifier Systems.

Keywords: Diversity, Correlation, Multi-Classifier System (MCS).

1 Introduction

The design of Multi-Classifier Systems (MCSs) is inspired by the group decision making process [13,14]. The motivation behind MCSs is that each classifier has its own strengths and weaknesses, and hence a group of classifiers could potentially leverage the wisdom of crowds. If each classifier in an MCS has expertise in classifying samples in some portions of a data space, the final output that is aggregated from all classifiers would become more reliable. More precisely, effective classifiers in an MCS are those that are accurate and independent. The former means that a classifier in an MCS is expected to provide performance at least better than random guessing, while the latter means that correlation between outputs of classifiers is expected to be small. This also implies that their outputs are expected to be diverse.

Diversity could be captured by disagreements between classifiers in an MCS and it plays a significant role in the success of MCSs [10]. However, the following research question becomes important for the design of MCSs: *Is there a relationship between diversity (between the member classifiers of an ensemble) and accuracy (of the ensemble)?* We address this research question by taking a different route and building a relationship between diversity and correlation, which could be related to accuracy.

Fig. 1 illustrates the focus of this paper. The relationship between diversity and accuracy is ambiguous in theory (e.g. *that elusive diversity* [9]). The relationship between correlation and the accuracy is relatively clear to researchers.

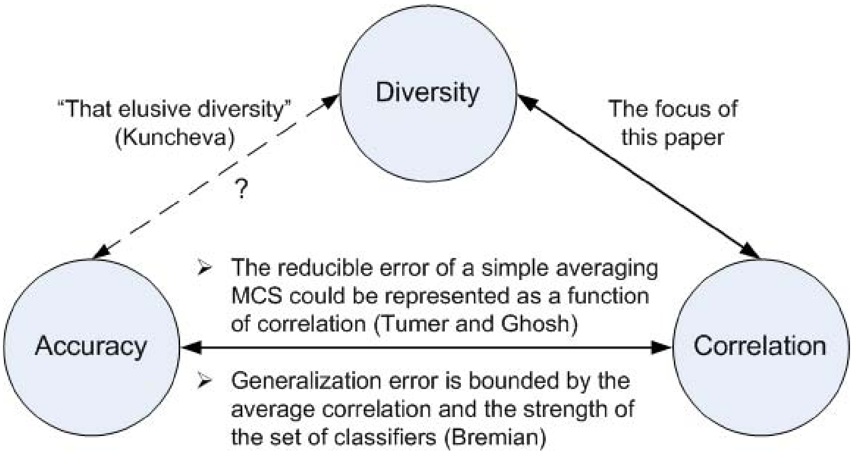

Diversity
"That elusive diversity" (Kuncheva)
?
The focus of this paper
Accuracy
Correlation
The reducible error of a simple averaging MCS could be represented as a function of correlation (Tumer and Ghosh)
Generalization error is bounded by the average correlation and the strength of the set of classifiers (Bremian)

Fig. 1. Relationships among the accuracy, diversity, and correlation

Fig. 1 also gives two more examples of such relationships: Tumer and Ghosh build such a relationship for simple averaging ensemble [15], while Bremian relates correlation to performance of Random Forests [1].

This paper provides a proxy to the relationship between diversity and accuracy, while it has a potential to assist with a design guideline for MCSs.

The rest of this paper is structured as follows. Section 2 gives a theoretical analysis and Section 3 discusses experimental results. Section 4 is a brief review of some related work, while Section 5 gives conclusions and future work.

2 Theoretical Analysis of Diversity and Correlation

Diversity has been studied by many researchers [3,6], but its relationship to accuracy is not clear. One difficulty is that there exists an elegant bias-variance-covariance decomposition framework for regression tasks, but the framework does not directly apply to classification tasks [4]. Here we do not directly connect diversity to accuracy. Rather we build a relationship between diversity and correlation.

Notations. For a set of N instances and two classifiers, N_{11} and N_{00} denote the numbers of instances for which both classifiers are correct and incorrect, respectively; N_{10} and N_{01} denote the numbers of instances for which only the first and the second classifier is correct, respectively. The following definitions are with respect to outputs of classifiers i and j.

Definition 1. *Disagreement measure (Dis) representing diversity [10].*

$$Dis_{i,j} = \frac{N_{01} + N_{10}}{N_{11} + N_{10} + N_{01} + N_{00}} = \frac{N_{01} + N_{10}}{N}$$

Definition 2. *Q-statistic or Q [10,11].*

$$Q_{i,j} = \frac{N_{11} \cdot N_{00} - N_{01} \cdot N_{10}}{N_{11} \cdot N_{00} + N_{01} \cdot N_{10}}$$

Definition 3. *Correlation [10].*

$$\rho_{i,j} = \frac{N_{11} \cdot N_{00} - N_{01} \cdot N_{10}}{\sqrt{(N_{11} + N_{10}) \cdot (N_{01} + N_{00}) \cdot (N_{11} + N_{01}) \cdot (N_{10} + N_{00})}}$$

One could calculate system-wise values by averaging all pairs, so we ignore the subscripts i and j for concise representation. Using these definitions and the inequality of arithmetic-geometric-harmonic means, we obtain Corollary 1, as given below.

Corollary 1. *Relationship between disagreement measure and Q-statistic.*

$$Q \leq \frac{(1 - Dis)^2 \cdot N^2 - 4 \cdot Dis \cdot N}{(1 - Dis)^2 \cdot N^2 + 4 \cdot Dis \cdot N}$$

Corollary 1 helps us connect diversity to correlation, since a connection between Definition 2 and Definition 3 is that the absolute value of correlation will be bounded by the absolute value of $Q - statistic$. Next we define $f(x)$ based on Corollary 1.

$$f(x) = \frac{(1 - x)^2 \cdot N^2 - 4 \cdot x \cdot N}{(1 - x)^2 \cdot N^2 + 4 \cdot x \cdot N}, \text{ where } x = Dis$$

Since $x = Dis$ and hence $x \in (0, 1)$, we have $f(0) = 1$ and $f(1) = -1$. As the goal is to have zero correlation, we would like to know the interception of $f(x)$ and x-axis. We call the interception the *critical value* of x (x_c) or the *critical point* of Dis, and the following *critical value* is straightforward:

$$x_c = (1 + \frac{2}{N}) - 2 \cdot \sqrt{\frac{1}{N} - \frac{1}{N^2}}$$

Before this critical point, higher diversity reduces correlation. This supports the intuition that higher diversity between classifiers is usually associated with a better MCS. When diversity crosses the critical point, increasing diversity would increase correlation while highly correlated classifiers usually correspond to an inferior MCS.

3 Experiments and Discussion

For each trial for a data set, we randomly draw samples and accordingly train a decision tree (without pruning). Similarly, we generate a disjoint set of samples and use it as a test set for each trial for a data set. To control the variable in, we create a dummy classifier for each decision tree. We repeat this 100 times and create 100 pairs of classifiers in every experiment, using the corresponding test set to evaluate each pair of classifiers, calculating values of disagreements, $Q - statistic$, and correlation. Figures in Appendix illustrate the results. Our findings are summarized as below:

- The relationship between disagreement measure and $Q-statistic$ is not linear. Although curves of the theoretical upper bounds do not always match curves of the observed values, they do indicate trends of curves of the observed values.
- For some data sets, the theoretical upper bounds of the values of $Q-statistic$ are close to the observed values. For all data sets, they are close when diversity is lower and especially when N is smaller.
- There are exceptions that are larger than the theoretical upper bounds corresponding to them. They are exceptional cases where $Q - statistic$ is 1.
- As N increases, curves move to the right. The critical point is a function of N. This suggests that we need to increase diversity in order to obtain low (or even 0) correlation when the number of training samples increases.
- It is not always the case that we observe critical points in experiments. For those showing critical points, we observe that $Q - statistic$ and correlation move away from 0 as the diversity increases. This follows our analysis.

Now we take a couple of steps further and use our analysis result to explain some interesting phenomenon. [7] showed theoretically that heterogeneity (i.e. using different algorithms in an MCS) would improve diversity among member classifiers in an MCS. Furthermore, [8] showed empirically that one could obtain such an improvement more often in bagging setting than in boosting setting; in addition it empirically showed that AdaBoost with heterogeneous algorithms would work better when the data set is larger. Compared to bagging, AdaBoost often provides higher diversity. When we introduce heterogeneity into AdaBoost, diversity will probably be increased. As discussed earlier, increasing diversity has positive effect in the left region (between 0 and the critical point) of the graph of $f(x)$, but it has negative effect in the right region (between the critical point and 1) of the graph of $f(x)$. Moreover, the smaller the data set, the smaller the critical point, the smaller the left region. Therefore, using heterogeneous algorithms in AdaBoost on small data sets may actually have negative effect to the performance.

4 Related Work

The importance of reducing correlation between classifiers in an MCS has been recognized [2]. Tumer and Ghosh discuss a framework that quantifies the need to reduce correlation between classifiers in an MCS, and associate the number of training samples (i.e. the size of the training set) with the effect of correlation reduction [15]. Our analysis suggests that, for example, the critical point of Dis depends on N. Mane et al. prove that classifiers trained by using independent feature sets give more independent estimations and their combination gives more accurate estimations [12].

The term anti-correlation is confusing. In [13] McKay and Abbass describe it as a mechanism to promote diversity, but they do not explain why anti-correlation is equivalent to diversity promoting. Our analysis, however, explains this: When we promote diversity to a certain level (i.e. we have diversity in the neighborhood of the critical point), we decrease the upper bound of the absolute value of correlation and thus it is possible to observe very low or even negative correlation.

In [5] Chung et al. argue that, given the average the accuracy (or performance) of classifiers, there is a linear relationship between correlation and disagreement measure. Nevertheless, our analysis clearly shows that the relationship is not linear and our experimental results do not reveal the linear relationship as given in [5].

5 Conclusions and Future Work

In this paper we explored the relationship between diversity, represented by disagreement, and correlation between classifiers in MCSs, conducting a theoretical analysis and experiments for the relationship between diversity and correlation. As a result, we demonstrated a nonlinear function for the relationship, while the experimental results reveal some interesting insights. Therefore, this paper contributes to a better understanding of the role of diversity in MCSs.

Future work includes (1) investigating a tighter theoretical bound of Q-statistic, (2) integrating our analysis into those proposed by others in order to build a more elegant relationship between diversity and accuracy, and (3) using our analysis result to assist with classifier selection and/or combination algorithms for MCSs.

Acknowledgements. The research reported herein was supported by the National Aeronautics and Space Administration via award number NNX08AC36A, by the National Science Foundation via award number CNS-0931931, and a gift from Huawei Telecom. We gratefully acknowledge all our sponsors. The findings presented do not in any way represent, either directly or through implication, the policies of these organizations.

References

1. Breiman, L.: Random Forests. Machine Learning 45(1), 5–32 (2001)
2. Brown, G., Wyatt, J., Tino, P.: Managing Diversity in Regression Ensembles. Journal of Machine Learning Research (JMLR) 6(September), 1621–1650 (2005)
3. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity Creation Methods: A Survey and Categorisation. Journal of Information Fusion 6(1), 5–20 (2005)
4. Brown, G.: Ensemble Learning. Encyclopedia of Machine Learning (2010)
5. Chung, Y., Hsu, D.F., Tang, C.Y.: On the Relationships Between Various Diversity Measures in Multiple Classifier Systems. In: International Symposium on Parallel Architectures, Algorithms, and Networks, pp. 184–190 (2008)
6. Ghosh, J.: Multiclassifier systems: Back to the future. In: Roli, F., Kittler, J. (eds.) MCS 2002. LNCS, vol. 2364, pp. 1–15. Springer, Heidelberg (2002)
7. Hsu, K.-W., Srivastava, J.: Diversity in Combinations of Heterogeneous Classifiers. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 923–932. Springer, Heidelberg (2009)
8. Hsu, K.-W., Srivastava, J.: An Empirical Study of Applying Ensembles of Heterogeneous Classifiers on Imperfect Data. In: Workshop on Data Mining When Classes are Imbalanced and Errors Have Costs (2009)
9. Kuncheva, I.: That Elusive Diversity in Classifer Ensembles. In: Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA), pp. 1126–1138 (2003)
10. Kuncheva, I., Whitaker, J.: Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. Machine Learning 51(2), 181–207 (2003)

11. Kuncheva, I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley Press, Chichester (2004)
12. Mane, S., Srivastava, J., Hwang, S.-Y.: Estimating missed actual positives using independent classifiers. In: International Conference on Knowledge Discovery and Data Mining (KDD), pp. 648–653 (2005)
13. McKay, R., Abbass, H.A.: Anti-correlation: A diversity promoting mechanism in ensemble learning. Australian Journal of Intelligence Information Processing Systems 7(3/4), 139–149 (2001)
14. Polikar, R.: Ensemble based systems in Decision making. IEEE Circuits and Systems Magazine 6(3), 21–45 (2006)
15. Tumer, K., Ghosh, J.: Error Correlation and Error Reduction in Ensemble Classifiers. Connection Science 8(3-4), 385–403 (1996)

Appendix A Experimental Results

In these figures, the x-axis is the value of disagreement measure (representing diversity) and y-axis corresponds to values of $Q - statistic$ or correlation ρ. A (blue) diamond and a (pink) square represent respectively an observed $Q - statistic$ and an observed correlation, while a (yellow) triangle gives an upper bound of the corresponding value of Q-statistic. We report results for 100 and 1000 training samples for each data set.

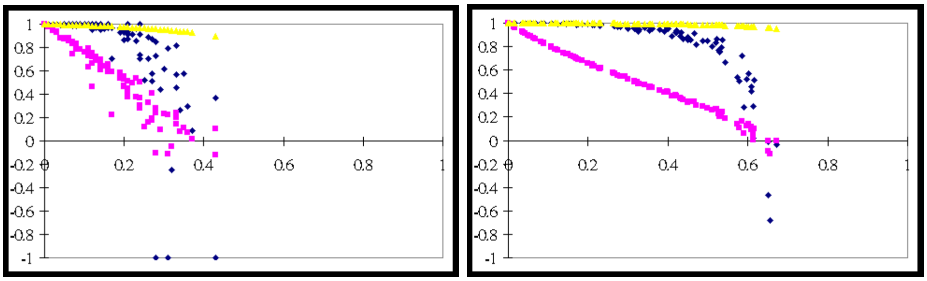


Fig. A1. Results for *Letter* with 100 (left) and 1000 (right) samples

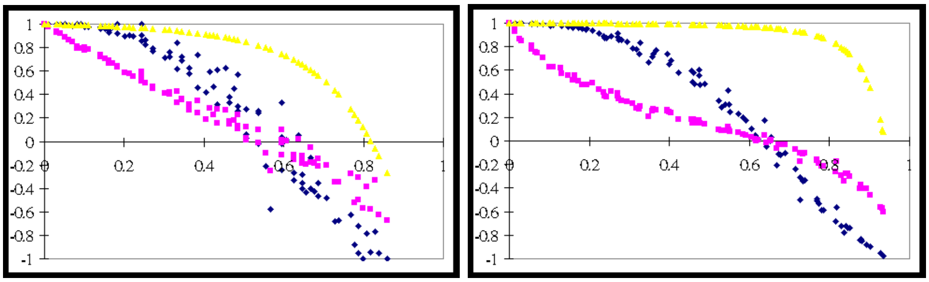


Fig. A2. Results for *Splice* with 100 (left) and 1000 (right) samples

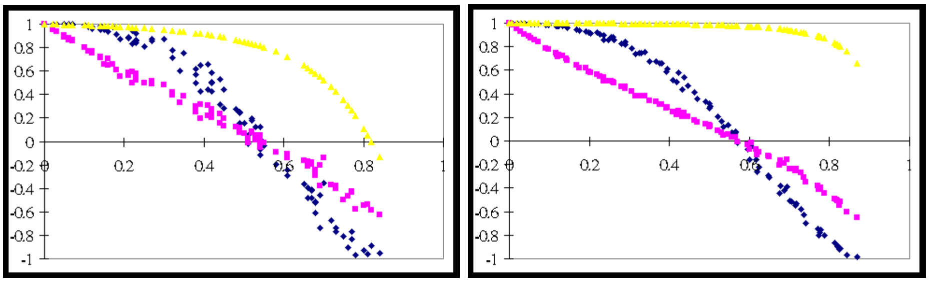


Fig. A3. Results for *Waveform-5000* with 100 (left) and 1000 (right) samples

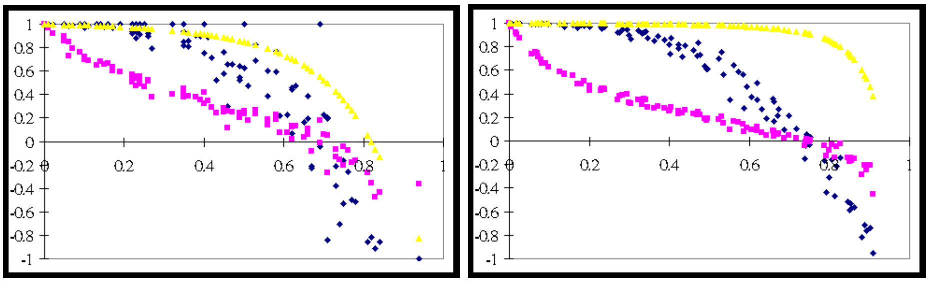


Fig. A4. Results for *Nursery* with 100 (left) and 1000 (right) samples

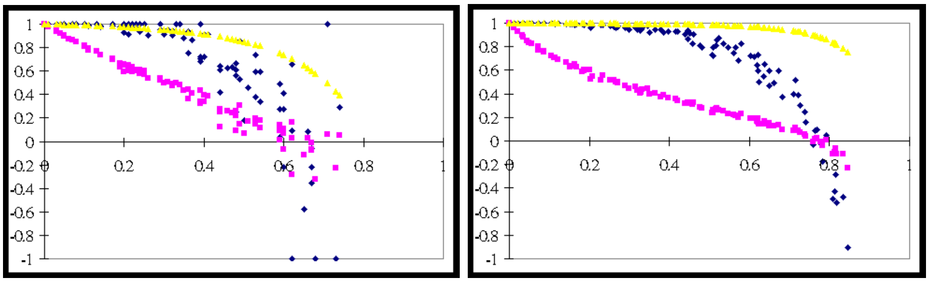


Fig. A5. Results for *Optdigits* with 100 (left) and 1000 (right) samples

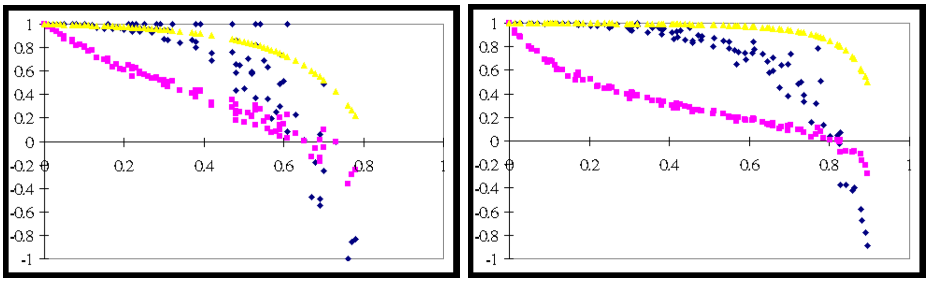


Fig. A6. Results for *Pendigits* with 100 (left) and 1000 (right) samples

Compact Margin Machine*

Bo Dai[1] and Gang Niu[2]

[1] NLPR/LIAMA, Institute of Automation, Chinese Academy of Science, Beijing 100190, P.R. China
bdai@nlpr.ia.ac.cn

[2] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, P.R. China
niugang@ai.nju.edu.cn

Abstract. How to utilize data more sufficiently is a crucial consideration in machine learning. Semi-supervised learning uses both unlabeled data and labeled data for this reason. However, Semi-Supervised Support Vector Machine (S3VM) focuses on maximizing margin only, and it abandons the instances which are not support vectors. This fact motivates us to modify maximum margin criterion to incorporate the global information contained in both support vectors and common instances. In this paper, we propose a new method, whose special variant is a semi-supervised extension of Relative Margin Machine, to utilize data more sufficiently based on S3VM and LDA. We employ *Concave-Convex Procedure* to solve the optimization that makes it practical for large-scale datasets, and then give an error bound to guarantee the classifier's performance theoretically. The experimental results on several real-world datasets demonstrate the effectiveness of our method.

1 Introduction

Semi-supervised learning paradigm, which blossoms out to utilize data sufficiently, has attracted more and more attention in machine learning community [1]. Semi-Supervised Support Vector Machine(S3VM) [2, 3], the semi supervised extension of support vector machine, is a state-of-the-art semi-supervised learning algorithm. It uses all the data no matter labeled or not to detect the margin and achieves significant improvement in practice. However, S3VM abandons the instances which are not support vectors. The framework [4] utilizes general unlabeled data, but still wasteful for non-support vectors, which stops them from further improving the performance.

To utilize the data more sufficiently and efficiently, we notice that compactness of projected data provides global information and thus is important for classification. Linear Discriminant Analysis [5] is a successful algorithm to utilize the information. Algorithm in [6] whitens the data when seeking decision boundary. Gaussian Margin Machine [7] controls the projected data compactness under a distribution assumption. Universum SVM [8] constrains range by

* This work is supported in part by Natural Science Foundation of China (No. 60275025).

data that is related but not belonged to any category. Another criterion that compresses the range of projected data is Relative Margin Machine[9] which is motivated from an affine invariance perspective and some probabilistic properties. Although these algorithms balance global and local information, these methods merely exploit labeled data but turn blind eyes to unlabeled data.

In this paper, we propose a method to use the information contained by instances sufficiently. After brief introductions to S3VM and LDA in Section 2, our framework incorporating LDA criterion with S3VM is presented in Section 3 and a relaxation of the criterion is given to make the optimization tractable. We also derive a special variant which can be viewed as the semi-supervised extension of Relative Margin Machine. The generalization bound is analyzed in Section 4. Finally, the experimental results are reported in Section 5.

2 Background

Suppose that we are given labeled data set $\mathcal{L} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)\}$ and unlabeled data set $\mathcal{U} = \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \ldots, \mathbf{x}_{l+u}\}$, both of which are drawn *i.i.d* from a certain data distribution $\mathcal{D}$, where $u \gg l$, $\mathbf{x}_i \in \mathbb{R}^d$ $(i = 1, 2, \ldots, l + u)$ and $y_j \in \{-1, 1\}, (j = 1, 2, \ldots, l)$ is the label of instance $\mathbf{x}_j$. The problem we want to solve is seeking a hypothesis $h : \mathbb{R}^d \to \{-1, +1\}$ which can classify the unlabeled data and unseen instances sampled from $\mathcal{D}$.

2.1 Semi-supervised SVM(S3VM)

Many semi-supervised methods find the suitable hypothesis by minimizing the criterion which utilize both labeled data and unlabeled data as

$$\min_f \|f\|_{\mathcal{H}_K} + C_1 \sum\nolimits_{i=1}^{l} \ell_1(\mathbf{x}_i, y_i; f) + C_2 \sum\nolimits_{i=l+1}^{l+u} \ell_2(\mathbf{x}_i; f) \tag{1}$$

where $\mathcal{H}$ is the reproducing kernel Hilbert space introduced by kernel function $\mathbb{K}$, ℓ_1 is a common classification loss function, and ℓ_2 is another loss function which utilizes unlabeled data only. S3VM employs *hinge loss* as ℓ_1 and *symmetric hinge loss* as ℓ_2. C_1 and C_2 are the parameters to balance the loss between labeled data, unlabeled data and function complexity. We take the set of linear form of $h = sign(f(\mathbf{x}))$, where $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b(\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R})$, as the hypothesis space. So the formulation of (1) could be transformed as follow:

$$\begin{aligned} &\min_{\mathbf{w},b,\eta,\xi\geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + C_1 \sum\nolimits_{i=1}^{l} \eta_i + C_2 \sum\nolimits_{i=l}^{l+u} \xi_i \\ &\text{s.t.} \quad y_i f(\mathbf{x}_i) \geq 1 - \eta_i, i = 1, \ldots, l, \quad |f(\mathbf{x}_i)| \geq 1 - \xi_i, i = l+1, \ldots, l+u. \end{aligned} \tag{2}$$

2.2 Linear Discriminative Analysis(LDA)

The basic principle of LDA is projecting the data into a subspace in which the instances in different categories can be scattered and the instances in the same category can be concentrated together. Let $\mathbf{m}_i = \frac{1}{n_i}\sum_{j \in \mathcal{C}_i} \mathbf{x}_j, i \in \{-1,+1\}$ is the sample mean of class $\mathcal{C}_i$ where n_i is the number of samples in $\mathcal{C}_i$, the mean of projected instances is given by $\hat{\mathbf{m}}_i = \frac{1}{n_i}\sum_{j \in \mathcal{C}_i} f(\mathbf{x}_j) = \mathbf{w}^T\mathbf{m}_i + b, i \in \{-1,+1\}$. So the distance between the projected means, *between-class distance*, is $\|\hat{\mathbf{m}}_{-1} - \hat{\mathbf{m}}_1\| = \|\mathbf{w}^T\mathbf{m}_{-1} - \mathbf{w}^T\mathbf{m}_1\|$. LDA seeks the largest *between-class distance* relative to total *within-class variance* which is defined by $\sum_{i=-1,+1}\mathbf{s}_i$ where $\mathbf{s}_i = \sum_{j \in \mathcal{C}_i}(f(\mathbf{x}_j) - \hat{\mathbf{m}}_i)^2$. The LDA criterion is defined as the ratio of the *between-class distance* to total *within-class variance*, $\mathcal{J}_\mathbf{w} = \frac{\|\hat{\mathbf{m}}_{-1} - \hat{\mathbf{m}}_1\|}{\sum_{i=-1,+1}\mathbf{s}_i}$.

3 Compact Margin Machine

The data compactness after projecting is very important in reflecting the data structure and can indeed help in classifying the data. It is necessary to restrict the range of projected data while seeking the largest margin. Within-class variance defined in LDA provides us a natural way to measure the projected data compactness. We modify $\mathbf{s}_i$ as $\sum_{j \in \mathcal{C}_i} \max\{0, |f(\mathbf{x}_j) - \hat{\mathbf{m}}_i| - \varepsilon\}, i = \{-1,+1\}$ instead of $2-norm$ form and introduce it as $\sum_{i=1}^{l+u} \ell_c(\mathbf{x}_i; \mathcal{L}, \mathcal{U}, f) = \sum_{i=-1,+1}\mathbf{s}_i$ to S3VM. We propose the model, *Compact Margin Machine*, as follow ($f(\mathbf{x})$ stands for $\mathbf{w}^T\mathbf{x} + b$):

$$\min_{\mathbf{w},b,\{\eta,\xi,\zeta\}\geq 0, d_i=\{0,1\}} \frac{1}{2}\|\mathbf{w}\|^2 + C_1\sum_{i=1}^{l}\eta_i + C_2\sum_{i=l+1}^{l+u}\xi_i + C_3\sum_{i=1}^{l+u}(\zeta_{1,i} + \zeta_{0,i}) \tag{3}$$
$$\text{s.t. } y_i f(\mathbf{x}_i) \geq 1 - \eta_i, i = 1,\ldots,l \quad |f(\mathbf{x}_i)| \geq 1 - \xi_i, i = l+1,\ldots,l+u$$

$$\frac{y_i+1}{2}\left|f(\mathbf{x}_i) - \frac{\sum_{i=l+1}^{l+u}(1-d_i)f(\mathbf{x}_i) + \sum_{i=1}^{l}(\frac{y_i+1}{2})f(\mathbf{x}_i)}{2u(1-r) + \sum_{i=1}^{l}(\frac{y_i+1}{2})}\right| \leq \varepsilon + \zeta_{0,i} \tag{4}$$

$$\frac{1-y_i}{2}\left|f(\mathbf{x}_i) - \frac{\sum_{i=l+1}^{l+u}d_i f(\mathbf{x}_i) + \sum_{i=1}^{l}(\frac{1-y_i}{2})f(\mathbf{x}_i)}{u(2r-1) + \sum_{i=1}^{l}(\frac{1-y_i}{2})}\right| \leq \varepsilon + \zeta_{1,i} \tag{5}$$

$$(1-d_i)\left|f(\mathbf{x}_i) - \frac{\sum_{i=l+1}^{l+u}(1-d_i)f(\mathbf{x}_i) + \sum_{i=1}^{l}(\frac{y_i+1}{2})f(\mathbf{x}_i)}{2u(1-r) + \sum_{i=1}^{l}(\frac{y_i+1}{2})}\right| \leq \varepsilon + \zeta_{0,i} \tag{6}$$

$$d_i\left|f(\mathbf{x}_i) - \frac{\sum_{i=l+1}^{l+u}d_i f(\mathbf{x}_i) + \sum_{i=1}^{l}(\frac{1-y_i}{2})f(\mathbf{x}_i)}{u(2r-1) + \sum_{i=1}^{l}(\frac{1-y_i}{2})}\right| \leq \varepsilon + \zeta_{1,i} \tag{7}$$

where we introduce the class balancing constraint $\frac{1}{u}\sum_{i=l+1}^{l+u} d_i = 2r - 1$ to avoid the trivial solution that assigns all the instances the same label [10]. We can use branch-and-bound algorithms to search the global optimal solution. However, the computational complexity is too high. We relax the constraints to make the optimization process easier and it can be proved that our relaxed constraints imply upper bounds of original loss functions.

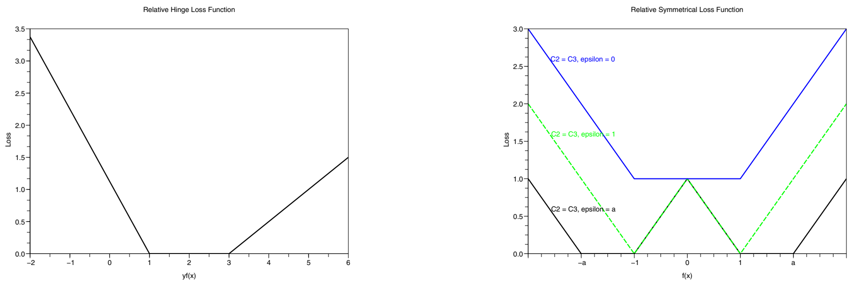


Fig. 1. Relaxed compact ε-hinge loss and relaxed compact ε-symmetric hinge loss

Proposition 1. *By replacing the constraints* (4)-(7) *with* $|\mathbf{w}^T\mathbf{x}_i+b| \leq \frac{1}{2}(\varepsilon+\zeta_i)$, *the loss will be no less than* ℓ_c *defined above.*

Proof. As the constraints (4)-(7) have the same form, without loss of generality, we consider the constraint (7) and the others are similar. Obviously, if there is a solution $\mathbf{w}$ that satisfies $|\mathbf{w}^T\mathbf{x}_i + b| \leq \frac{1}{2}(\varepsilon + \zeta_i)$, then we have

$$d_i\big|\big((\mathbf{w}^T\mathbf{x}_i + b) - \frac{\sum_{j=l+1}^{l+u} d_j(\mathbf{w}^T\mathbf{x}_j + b) + \sum_{j=1}^{l}(\frac{1-y_j}{2})(\mathbf{w}^T\mathbf{x}_j + b)}{\sum_{j=l}^{l+u} d_j + \sum_{j=1}^{l}(\frac{1-y_j}{2})}\big)\big| \leq$$
$$d_i\big(|\mathbf{w}^T\mathbf{x}_i + b| + \big|\frac{\sum_{j=l+1}^{l+u} d_j(\mathbf{w}^T\mathbf{x}_j + b) + \sum_{j=1}^{l}(\frac{1-y_j}{2})(\mathbf{w}^T\mathbf{x}_j + b)}{\sum_{j=l}^{l+u} d_j + \sum_{j=1}^{l}(\frac{1-y_j}{2})}\big|\big) \leq$$
$$|\mathbf{w}^T\mathbf{x}_i + b| + \frac{1}{n_j}\sum_{j\in\mathcal{C}_{y_i}} |\mathbf{w}^T\mathbf{x}_j + b| \leq \varepsilon + \frac{1}{2}\zeta_i + \frac{1}{2n_j}\sum_{j\in\mathcal{C}_{y_i}} \zeta_j$$

The inequality is derived from triangle inequality. We obtain that the relaxed constraints provide an upper bound and $\sum_{i=1}^{n} \ell_c(\mathbf{x}_i; \mathcal{L}, \mathcal{U}, f) \leq \sum_{i=1}^{n} \zeta_i$. □

Proposition 1 suggests that the relaxed constrains are helpful in reflecting the projected data compactness. Mathematically, CMM is relaxed as:

$$\begin{aligned}
\min_{\mathbf{w},b,\{\eta,\xi,\zeta\}\geq 0} & \frac{1}{2}\|\mathbf{w}\|^2 + C_1\sum_{i=1}^{l}\eta_i + C_2\sum_{i=l+1}^{l+u}\xi_i + C_3\sum_{i=1}^{l+u}\zeta_i \\
\text{s.t.} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \eta_i, \quad i = 1, \ldots, l \\
& |\mathbf{w}^T\mathbf{x}_i + b| \geq 1 - \xi_i, \quad i = l+1, \ldots, l+u \\
& |\mathbf{w}^T\mathbf{x}_i + b| \leq \varepsilon + \zeta_i, \quad i = 1, \ldots, l+u
\end{aligned} \tag{8}$$

Note that relaxed constraints modify both of the loss functions for labeled and unlabeled data given by S3VM actually. The loss function for the labeled data is $\ell_1^{'} = \max\{0, 1 - yf(x)\} + \frac{C_3}{C_1}\max\{0, |f(x)| - \varepsilon\}$ which we named as *relaxed compact ε-hinge loss.* On the other hand, the loss function of the unlabeled data is adapted as $\ell_2' = \max\{0, 1 - |f(x)|\} + \frac{C_3}{C_2}\max\{0, |f(x)| - \varepsilon\}$, named as *relaxed compact ε-symmetric hinge loss*. The loss functions are shown above. It is not easy to solve the optimization problem (8) directly because of the non-convex property of *relaxed compact ε-symmetric hinge loss*. As [2], Mix Integer Programming can find the global optimal solution of (8). However, the computational complexity of MIP is usually very high. By employing *Concave-Convex Procedure(CCCP)* [11], (8) can be solved effciently [12]. We set $\mathbf{x}_i = \mathbf{x}_{i-u}, i = l+u+1, \ldots, l+2u$, $y_i = 1, i = l+1, \ldots, l+u$, $y_i = -1, i = l+u+1, \ldots, l+2u$ and rewrite the relaxed Compact Margin Machine criterion (8) as the sum of a convex part

$$\begin{aligned}\mathcal{J}_{vex} = \tfrac{1}{2}\|\mathbf{w}\|^2 + C_1\textstyle\sum_{i=1}^{l}\max\{0, 1 - y_i f(\mathbf{x}_i)\} + C_2\textstyle\sum_{i=l+1}^{l+2u}\max\{0, 1 - y_i f(\mathbf{x}_i)\}\\ + C_3\textstyle\sum_{i=1}^{l+2u}\max\{0, |f(\mathbf{x}_i)| - \varepsilon\}\end{aligned}$$

and a concave part $\mathcal{J}_{cav} = -C_2\sum_{i=l+1}^{l+2u}\max\{0, \delta - y_i f(\mathbf{x}_i)\}$ where $\delta \in (-1, 0]$.

```
1  Initialize $\boldsymbol{\theta}^0 = (\mathbf{w}^0, b^0)$ with a standard SVM solution on labeled data
2  Set
       $\beta_i = \begin{cases} C_2 & \text{if } y_i f_{\theta^0}(\mathbf{x}_i) < \delta \quad \textit{and} \quad i \geq l+1 \\ 0 & \text{otherwise} \end{cases}$
   while $\boldsymbol{\beta}^{t+1} \neq \boldsymbol{\beta}^t$ do
3      solve the convex problem where $\mathbb{K}$ is kernel
       $\min_{\boldsymbol{\alpha},\boldsymbol{\gamma},\hat{\boldsymbol{\gamma}}\geq 0} \frac{1}{2}\big((\boldsymbol{\alpha}-\boldsymbol{\beta})\odot\mathbf{y}-\boldsymbol{\gamma}+\hat{\boldsymbol{\gamma}}\big)^T\mathbb{K}\big((\boldsymbol{\alpha}-\boldsymbol{\beta})\odot\mathbf{y}-\boldsymbol{\gamma}+\hat{\boldsymbol{\gamma}}\big)-\boldsymbol{\alpha}^T\mathbf{1}+\varepsilon\boldsymbol{\gamma}^T\mathbf{1}+\varepsilon\hat{\boldsymbol{\gamma}}^T\mathbf{1}$
       subject to $(\boldsymbol{\beta}-\alpha)^T\mathbf{y}+\boldsymbol{\gamma}^T\mathbf{1}-\hat{\boldsymbol{\gamma}}^T\mathbf{1} = 0, \hat{\boldsymbol{\gamma}}+\boldsymbol{\gamma} \leq C_3\mathbf{1}$
           $\alpha_i \leq C_1, i = 1, \ldots, l, \alpha_i \leq C_2, i = 1+1, \ldots, l+2u$
4      compute $b^{t+1}$ by $f_{\boldsymbol{\theta}^{t+1}}(\mathbf{x}_i) = \big((\boldsymbol{\alpha}-\boldsymbol{\beta})\odot\mathbf{y}-\boldsymbol{\gamma}+\hat{\boldsymbol{\gamma}}\big)^T\mathbb{K}_{i\bullet} + b^{t+1}$
5      compute $y_i f_{\boldsymbol{\theta}^{t+1}}(\mathbf{x}_i), i = l+1, \ldots, l+2u$ by $KKT$ conditions
6      alternate $\boldsymbol{\beta}$
           $\beta_i = \begin{cases} C_2 & \text{if } y_i f_{\theta^{t+1}}(\mathbf{x}_i) < \delta \quad \textit{and} \quad i \geq l+1 \\ 0 & \text{otherwise} \end{cases}$
7  end
```

Algorithm 1. Concave-Convex Procedure for Relaxed CMM

At each iteration, *CCCP* solves $\min_{\mathbf{w},b}\mathcal{J}_{vex}(\mathbf{w}, b) + \mathcal{J}_{cav}'(\mathbf{w}^t, b^t)\cdot(\mathbf{w}, b)$ until convergence. The convergence of *CCCP* has been shown by [13]. The steps of algorithm are shown in Algorithm 1.

3.1 Special Variant and the Relationship to RMM

In this section, we derive a special variant from (8) which can be solved more efficiently. We set the parameters $C_2 = C_3$, $\varepsilon = 0$ in (8), so the loss function of unlabeled data is the blue one in Fig.1. The mathematical form is as following:

$$\begin{aligned} \min_{\mathbf{w},b,\{\eta,\zeta\}\geq 0} & \frac{1}{2}\|\mathbf{w}\|_2^2 + C_1\sum_{i=1}^{l}\eta_i + C_2\sum_{i=1}^{l+u}(\zeta_i+\hat{\zeta}_i) \\ \text{s.t. } & y_i(\mathbf{w}^T\mathbf{x}_i+b)\geq 1-\eta_i, \quad i=1,\dots,l \\ & \mathbf{w}^T\mathbf{x}_i+b\leq\varepsilon+\zeta_i, -(\mathbf{w}^T\mathbf{x}_i+b)\leq\varepsilon+\hat{\zeta}_i, \quad i=1,\dots,l+u \end{aligned} \tag{9}$$

This special variant of relaxed Compact Margin Machine utilizes the labeled data to control the margin and both the labeled and unlabeled data to compress the range of projected data. It can be viewed as a semi-supervised extension of RMM. RMM introduces the relaxed constraints (8) from the other motivations such as affine invariance perspective and some probabilistic properties. From this aspect, we can conclude that our model achieves the same properties that are given by RMM.

4 Theoretical Analysis

In this section, we derive the empirical transductive Rademacher complexity [14] for function classes relaxed Compact Margin Machine which can be plugged into uniform Rademacher error bound directly.

Firstly, we define the function class of S3VM as $\mathscr{H}_{\mathcal{D}} = \{\mathbf{w}^T\mathbf{x} \mid \frac{1}{2}\mathbf{w}^T\mathbf{w}\leq\mathcal{D}\}$ and the function class of *relaxed Compact Margin Machine* as $\mathscr{F}_{\mathcal{D},\mathcal{C}_3} = \{\mathbf{w}^T\mathbf{x} \mid \frac{1}{2}\mathbf{w}^T\mathbf{w}+\frac{\mathcal{C}_3}{2}\|\mathbf{w}^T\mathbf{z}_i\|_2^2\leq\mathcal{D}, 1\leq i\leq n\}$ where $\mathscr{Z}=\{\mathbf{z}_1,\dots,\mathbf{z}_n\}$ is an extra dataset drawn from the same distribution for convenience of proof. However, in practice, we may use training dataset and testing dataset as the extra dataset. We can derive the empirical transductive Rademacher complexity of relaxed CMM and S3VM.

Lemma 2. *The transductive Rademacher complexity of* Semi-Supervised SVM $R_{l+u}(\mathscr{H}_{\mathcal{D}})\leq 2\sqrt{\frac{D}{lu}\sum_{i=1}^{r}\lambda_i}$*, where* $\{\lambda_i\}_{i=1}^{r}$ *are the singular eigenvalues of Gram matrix of the data.* □

Lemma 3. *The transductive Rademacher complexity of* relaxed Compact Margin Machine $R_{l+u}(\mathscr{F}_{\mathcal{D},\mathcal{C}_3})\leq \min_{\alpha\geq 0}\frac{1}{l+u}\sum_{i=1}^{l+u}\mathbf{x}_i\mathbb{K}_{\alpha,\mathcal{C}_3}\mathbf{x}_i + \frac{lu}{(m+u)^2}D\sum_{i=1}^{n}\alpha_i$*, where* $\mathbb{K}_{\alpha,C_3}=\sum_{i=1}^{n}\alpha_i\mathbf{I}+\mathcal{C}_3\sum_{i}^{n}\alpha_i\mathbf{z}_i\mathbf{z}_i^T$*.* □

Based on theorems in [14], the following corollary provides an upper bound on the error rate:

Corollary 4. *Fix* $\gamma>0$*, let* $\mathscr{F}$ *be the set of function. Let* $c_0=\sqrt{\frac{32\ln 4e}{3}}$ *and* $Q=\frac{l+u}{lu}$*. Then with probability at least* $1-\delta$ *over the training set, the following bound holds:*

$$P[y\neq sign(f(x))]\leq\frac{1}{n\gamma}\sum_{i=1}^{l}\xi_i+\frac{R_{l+u}(\mathscr{F})}{\gamma}+c_0Q\sqrt{\min\{l,u\}}+\sqrt{2Q\ln\frac{1}{\delta}}$$

where $\xi_i=\max\{0,1-y_if(x_i)\}$*.*

Table 1. Experimental Results (Accuracy in percentage)

Performance Comparison on MNIST (mean%)								
num	#label/class	5	10	20	40	50	100	200
400	SVM	71.19	89.76	92.36	94.12	94.41	95.32	95.50
	LDA	84.31	89.34	92.13	93.16	92.85	93.23	93.84
	RMM	84.62	89.90	92.58	94.43	94.81	95.95	**96.13**
	TSVM	87.18	93.54	**95.11**	95.15	95.20	95.23	95.50
	LapSVM	**88.83**	91.68	93.22	94.52	94.63	95.11	95.50
	CMM	88.29	**93.80**	95.07	**95.60**	**95.76**	**96.10**	**96.13**
600	TSVM	90.32	93.59	94.76	95.20	95.26	95.72	96.22
	LapSVM	**91.23**	92.31	93.50	94.33	94.69	95.39	96.47
	CMM	90.73	**94.62**	**95.27**	**95.83**	**96.01**	**96.33**	**96.72**
800	TSVM	**94.52**	93.86	94.65	95.41	95.45	96.12	96.42
	LapSVM	92.13	93.44	94.13	95.00	95.20	95.87	96.67
	CMM	91.37	**94.99**	**95.70**	**96.10**	**96.31**	**96.63**	**96.89**

Performance Comparison on TDT2 (mean%)					
num	#label/class	5	15	50	150
300	SVM	87.09	93.62	96.38	98.00
	LDA	93.44	96.33	94.45	94.74
	RMM	94.39	96.84	97.60	**98.55**
	TSVM	**96.48**	95.95	97.73	98.00
	LapSVM	89.12	95.79	97.52	98.00
	CMM	94.90	**97.72**	**98.52**	**98.55**
600	TSVM	**96.23**	95.63	96.11	97.11
	LapSVM	88.50	93.24	95.89	97.69
	CMM	95.45	**97.20**	**97.38**	**97.78**

5 Experiment

The proposed algorithm is evaluated on two benchmark datasets, MNIST[1] and TDT2[2], to illustrate the effectiveness. We compare our model with SVM, Kernel LDA, RMM, LapSVM and S3VM. We implement our algorithm, RMM and LapSVM based on CVX[3]. The SVM and S3VM are solved by SVM^{light} [4]. Among all the experiments, we select the best kernel by 10-fold cross-validation. The other parameters are also tuned beforehand. We chose the digits “8” vs “9” from MNIST dataset for binary classification problem because these two digits are difficult to discriminate visually. In TDT2 dataset, we chose categories randomly. We conduct experiments on several different amount of labeled and unlabeled data. The final test accuracy is given as the average of 10 independent trials on the test dataset.

[1] The MNIST dataset can be download from `http://yann.lecun.com/exdb/mnist/`
[2] The TDT2 dataset can is available at `http://www.nist.gov/speech/tests/tdt/tdt98`
[3] The CVX matlab code be obtained from `http://www.stanford.edu/~boyd/cvx/`
[4] The package of SVM^{light} can be download from `http://svmlight.joachims.org/`

The training data are selected randomly each time. We further select a certain number of labeled data from the selected training dataset randomly.

The weakness of large margin criterion can be observed from the results. When labeled data are rare, the algorithms that compress the range of projected data, such as LDA and RMM, perform better than SVM. Our semi-supervised algorithm achieves significant improvement on the two datasets with benefit from the compact margin. Fortunately, our algorithm suffers the slightest depression compared with others when the unlabeled data hurt the classifiers.

6 Conclusion

We propose a novel semi-supervised algorithm which can utilize the data sufficiently. To make our method capable to handle large-scale applications, we employ Concave-Convex Procedure to solve the non-convex problem. A semi-supervised extension of RMM can be derived from our model. Moreover, we provide theoretical analyses to guarantee the performance of our algorithm, and finally experimental results show that the proposed algorithm improves the accuracy. A efficient global optimization method for exact solution is our future direction.

References

[1] Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-Supervised Learning. MIT Press, Cambridge (2006)

[2] Bennett, K.P., Demiriz, A.: Semi-supervised support vector machines. In: 12th NIPS, pp. 368–374 (1998)

[3] Joachims, T.: Transductive inference for text classification using support vector machines. In: 16th ICML, pp. 200–209 (1999)

[4] Huang, K., Xu, Z., King, I., Lyu, M.R.: Semi-supervised learning from general unlabeled data. In: Perner, P. (ed.) ICDM 2008. LNCS (LNAI), vol. 5077, pp. 273–282. Springer, Heidelberg (2008)

[5] Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annals Eugen. 7, 179–188 (1936)

[6] Xiong, R., Cherkassky, V.: A combined svm and lda approach for classification. In: IJCNN, pp. 157–171 (2005)

[7] Crammer, K., Mohri, M., Pereira, F.: Gaussian margin machines. In: 12th AISTATS (2009)

[8] Weston, J., Collobert, R., Sinz, F., Bottou, L., Vapnik, V.: Inference with the universum. In: 23rd ICML, pp. 1009–1016 (2006)

[9] Shivaswamy, P., Jebara, T.: Relative margin machines. In: 22nd NIPS (2008)

[10] Chapelle, O., Sindhwani, V., Keerthi, S.S.: Optimization techniques for semi-supervised support vector machines. JMLR 9, 203–233 (2008)

[11] Yuille, A.L., Rangarajan, A.: The concave-convex procedure (cccp). In: 15th NIPS. MIT Press, Cambridge (2001)

[12] Collobert, R., Sinz, F., Weston, J., Bottou, L.: Large scale transductive svms. JMLR 7, 1687–1712 (2006)

[13] Sriperumbudur, B., Lanckriet, G.: On the convergence of the concave-convex procedure. In: 23rd NIPS (2009)

[14] El-Yaniv, R., Pechyony, D.: Transductive rademacher complexity and its applications. In: 20th COLT, pp. 157–171 (2007)

Author Index