Johannes Fürnkranz
Eyke Hüllermeier
Tomoyuki Higuchi (Eds.)

LNAI 8140

Discovery Science

16th International Conference, DS 2013
Singapore, October 2013
Proceedings

Springer

Johannes Fürnkranz Eyke Hüllermeier
Tomoyuki Higuchi (Eds.)

Discovery Science

16th International Conference, DS 2013
Singapore, October 6-9, 2013
Proceedings

Springer

Volume Editors

Johannes Fürnkranz
TU Darmstadt, Germany
E-mail: fuernkranz@informatik.tu-darmstadt.de

Eyke Hüllermeier
Phillips-Universität Marburg, Germany
E-mail: eyke@mathematik.uni-marburg.de

Tomoyuki Higuchi
The Institute of Statistical Mathematics, Tokyo, Japan
E-mail: higuchi@ism.ac.jp

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-40896-0 e-ISBN 978-3-642-40897-7
DOI 10.1007/978-3-642-40897-7
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013947599

CR Subject Classification (1998): I.2, H.3, H.4, H.2.8, J.1, H.5, C.2

LNCS Sublibrary: SL 7 – Artificial Intelligence

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

Printed on acid-free paper

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

Preface

This year's International Conference on Discovery Science (DS) was the 16th event in this series. Like in previous years, the conference was co-located with the International Conference on Algorithmic Learning Theory (ALT), which is already in its 24th year. Starting in 2001, ALT/DS is one of the longest-running series of co-located events in computer science. The unique combination of recent advances in the development and analysis of methods for automatic scientific knowledge discovery, machine learning, intelligent data analysis, and their application to knowledge discovery on the one hand, and theoretical and algorithmic advances in machine learning on the other hand makes every instance of this joint event unique and attractive.

This volume contains the papers presented at the 16th International Conference on Discovery Science, while the papers of the 24th International Conference on Algorithmic Learning Theory are published in a companion volume edited by Sanjay Jain, Rémi Munos, Frank Stephan, and Thomas Zeugmann (Springer LNCS Vol. 8139). We had the pleasure of selecting contributions from 52 submissions by 142 authors from 23 countries. Each submission was reviewed by three Program Committee members. The program chairs eventually decided to accept 23 papers, yielding an acceptance rate of slightly less than 45%.

The program also included 3 invited talks and 2 tutorials. In the joint DS/ALT invited talk, Nir Ailon gave a presentation about "Learning and Optimizing with Preferences." The DS invited talk by Hannu Toivonen was on "Creative Computers and Data Mining." Finally, DS participants also had the opportunity to attend the ALT invited talk on "Efficient Algorithms for Combinatorial Online Prediction", which was given by Eiji Takimoto. The two tutorial speakers were Krzysztof Dembczyński ("Multi-Target Prediction") and Nader H. Bshouty ("Exact Learning from Membership Queries: Some Techniques, Results and New Directions").

This year, both conferences were held in Singapore, organized by the School of Computing, National University of Singapore (NUS). We are very grateful to the School of Computing at NUS for sponsoring the conferences and providing administrative support. In particular, we thank the local arrangement chair, Lee Wee Sun, and his team, Mark Bartholomeusz and Kee Yong Ngee, as well as all of the other administrative staff at the School of Computing, NUS, for their efforts in organizing the two conferences. We would like to thank the Office of Naval Research Global for the generous financial support provided under ONRG GRANT N62909-13-1-C208.

We would also like to thank all authors of submitted papers, the Program Committee members, and the additional reviewers for their efforts in evaluating the submitted papers, as well as the invited speakers and tutorial presenters. We are grateful to Frank Stephan and Sanjay Jain for their timely answers to many questions and for ensuring a smooth coordination with ALT, Thomas Zeugmann for his help with the proceedings, Robin Senge for putting up and maintaining our website, and Andrei Voronkov for making EasyChair freely available. Finally, special thanks go to the Discovery Science Steering Committee, in particular to its past and current chairs, Einoshin Suzuki and Akihiro Yamamoto, for entrusting us with the organization of the scientific program of this prestigious conference.

July 2013

Johannes Fürnkranz
Eyke Hüllermeier
Tomoyuki Higuchi

Organization

Conference Chair

Tomoyuki Higuchi	The Institute of Statistical Mathematics, Tokyo

Program Chairs

Johannes Fürnkranz	TU Darmstadt, Germany
Eyke Hüllermeier	University of Marburg, Germany

Local Arrangements Chair

Wee Sun Lee

Local Organization Team

Frank Stephan
Sanjay Jain
Mark Bartholomeusz
Yong Ngee Kee
Rachel Goh
Noraiszah Bte Hamzah

Program Committee

Fabio Aiolli	Università di Padova, Italy
Hiroki Arimura	Hokkaido University, Japan
Hideo Bannai	Kyushu University, Japan
Michael Berthold	University of Konstanz, Germany
Albert Bifet	University of Waikato, New Zealand
Hendrik Blockeel	Katholieke Universiteit Leuven, Belgium
Henrik Boström	Stockholm University, Sweden
Jean-Francois Boulicaut	INSA Lyon, France
Ulf Brefeld	TU Darmstadt, Germany
Robert Busa-Fekete	University of Marburg, Germany
Weiwei Cheng	University of Marburg, Germany
Bruno Cremilleux	University of Caen, France
Luc De Raedt	Katholieke Universiteit Leuven, Belgium
Juan José Del Coz	University of Oviedo at Gijón, Spain
Krzysztof Dembczyński	Poznan University of Technology, Poland

Additional Reviewers

Alatrista Salas, Hugo
Appice, Annalisa
Bothorel, Cecile
Cerf, Loïc
Desmier, Elise
Hoang, Lam Thanh
Huang, Sheng-Jun
Kaytoue, Mehdi
Kocev, Dragi
Kuboyama, Tetsuji
Kuzmanovski, Vladimir
Moreira-Matias, Luis
Otaki, Keisuke
Panov, Pance
Phan, Nhat Hai
Rabatel, Julien
Ribeiro, Rita
Sakamoto, Hiroshi
Soulet, Arnaud
Tromp, Erik
Zenko, Bernard

Invited Talks
(Abstracts)

Learning and Optimizing with Preferences[*]

Nir Ailon

Department of Computer Science
Technion Israel Institute of Technology
Haifa, Israel
nailon@cs.technion.ac.il

Abstract. Preferences and choices are a central source of information generated by humans. They have been studied for centuries in the context of social choice theory, econometric theory, statistics and psychology. At least two Nobel prizes in economics have been awarded for work reasoning about human preferences and choices.

In the last two decades computer scientists have studied preference data, which became available in unprecedented quantities: Each time we click or tap on a search result, a sponsored ad or a product recommendation, we express preference of one alternative from a small set of alternatives. Additionally, many crowsdsourcing systems explicitly ask (paid?) experts to solicit preferences or even full rankings of alternative sets.

What are the advantages of preferences compared to other forms of information, and what challenges do they give rise to? I will present important problems and survey results.

[*] An extended version of this paper can be found in Sanjay Jain, Rémi Munos, Frank Stephan, and Thomas Zeugmann, *Proceedings of the 24th International Conference on Algorithmic Learning Theory (ALT-13)*, Lecture Notes in Computer Science Vol. 8139, Springer-Verlag, 2013.

Efficient Algorithms for Combinatorial Online Prediction[*]

Eiji Takimoto

Department of Informatics
Kyushu University
Japan
eiji@inf.kyushu-u.ac.jp

We study online linear optimization problems over combinatorial concept classes $\mathcal{C} \subset \mathbb{R}^n$ that are defined in some combinatorial ways. Examples of such classes are *s*-*t* paths in a given graph, spanning trees of a given graph, permutations over a given set, truth assignments for a given CNF formula, set covers of a given subset family, and so on. Typically, those concept classes are finite but contain exponentially many concepts. The problem for a concept class $\mathcal{C}$ is described as follows: At each trial t, the algorithm chooses a concept $\boldsymbol{c}_t \in \mathcal{C}$, the adversary returns a loss vector $\boldsymbol{\ell}_t \in [0,1]^n$, and the algorithm incurs a loss given by $\boldsymbol{c}_t \cdot \boldsymbol{\ell}_t$. The goal of the algorithm is to make the cumulative loss not much larger than that of the best concept in $\mathcal{C}$.

One of the major approaches to the problem is to apply Follow the Regularized Leader (FTRL) framework, in which two external procedures *projection* and *decomposition* are assumed to be implemented. In other words, for each concept class $\mathcal{C}$, we need to design algorithms for the two procedures. In this talk, we give a projection and decomposition algorithms that work efficiently and uniformly for a wide class of concept classes. More precisely, if the convex hull of $\mathcal{C}$ is a submodular base polyhedron specified by a submodular functoin f, then the two procedures are computed in polynomial time, assuming that f can be computed in polynomial time.

Another approach is to use an offline algorithm as an oracle to construct an online algorithm. Here, the offline algorithm solves the corresponding offline optimization problem. Follow the perturbed leader (FPL) and the Online Frank-Wolfe (OFW) are of this type. In this talk, we consider a harder but typical case where the offline optimization problem for $\mathcal{C}$ is NP-hard, for which none of the FTRL, FPL and OFW work. The FTRL has been generalized so that it works when an offline *approximation* algorithm is available. However, it is not efficient enough. In this talk, we give a more efficient online algorithm using an offline approximation algorithm which has a guarantee of a certain integrity gap.

[*] An extended version of this paper can be found in Sanjay Jain, Rémi Munos, Frank Stephan, and Thomas Zeugmann, *Proceedings of the 24th International Conference on Algorithmic Learning Theory (ALT-13)*, Lecture Notes in Computer Science Vol. 8139, Springer-Verlag, 2013.

Creative Computers and Data Mining

Hannu Toivonen

Department of Computer Science and HIIT
University of Helsinki
Finland
hannu.toivonen@cs.helsinki.fi

Abstract. In the field of computational creativity, researchers aim to give computers creative skills, such as those needed in writing poetry or composing music. Obviously, an agent needs to know the field in which it operates. This is where data mining has great potential: making creative agents adaptive to various fields and genres by automatic discovery of relevant information from existing creative artifacts. We give several examples of how verbal creativity can benefit from data mining of existing text corpora.

On the other hand, computational creativity tools allow a whole new approach to data analysis. In this "Affective Data Analysis", the goal is to turn data into a subjective, esthetic experience by automatic or semiautomatic creation of a novel artifact using the user's data as inspiration. This is in strong contrast with traditional data analysis methods that emphasize cold facts instead of warm feelings. We illustrate this idea with musicalization of sleep measurements and chat discussions.

Table of Contents

Mixture Models from Multiresolution 0-1 Data

Prem Raj Adhikari and Jaakko Hollmén

Helsinki Institute for Information Technology (HIIT), and
Department of Information and Computer Science (ICS)
Aalto University School of Science,
PO Box 15400, FI-00076 Aalto, Espoo, Finland
{prem.adhikari,jaakko.hollmen}@aalto.fi

Abstract. Multiresolution data has received considerable research interest due to the practical usefulness in combining datasets in different resolutions into a single analysis. Most models and methods can only model a single data resolution, that is, vectors of the same dimensionality, at a time. This is also true for mixture models, the model of interest. In this paper, we propose a multiresolution mixture model capable of modeling data in multiple resolutions. Firstly, we define the multiresolution component distributions of mixture models from the domain ontology. We then learn the parameters of the component distributions in the Bayesian network framework. Secondly, we map the multiresolution data in a Bayesian network setting to a vector representation to learn the mixture coefficients and the parameters of the component distributions. We investigate our proposed algorithms on two data sets. A simulated data allows us to have full data observations in all resolutions. However, this is unrealistic in all practical applications. The second data consists of DNA aberrations data in two resolutions. The results with multiresolution models show improvement in modeling performance with regards to the likelihood over single resolution mixture models.

Keywords: Multiresolution data, Mixture Models, Bayesian Networks.

1 Introduction

A phenomenon or a data generating process measured in different levels of accuracy results in multiresolution data. This difference in accuracy arises because of improvement in measurement technology [1]. Newer generation technology measures finer units of data producing data in fine resolution. In contrast, older generation technology measures only the coarse units of data producing data in coarse resolution. Thus, accumulation of data over long duration of time results in multiresolution data. The availability of multiresolution data ranges across diverse application domains such as computer vision, signal processing, telecommunications, and biology [2].

The domain of scale-space theory [4], and wavelets [5] have close affinity with the domain of multiresolution modeling thus widening the scope of multiresolution modeling research. Furthermore, multiresolution data falls under one of the

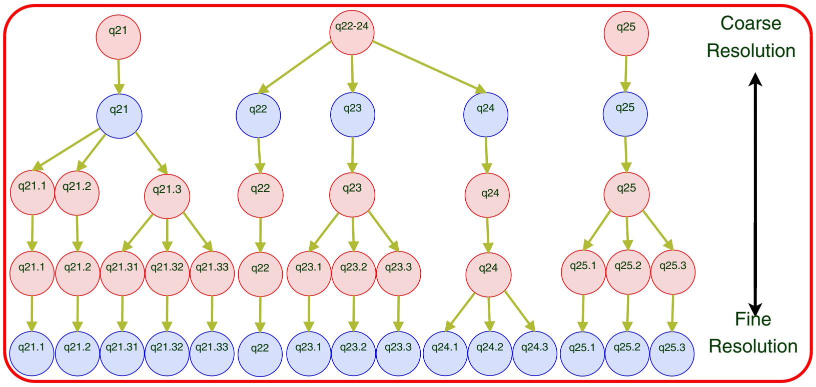


Fig. 1. A typical dichotomy of universals and particulars in giving rise to multiresolution data. Figure shows a part of chromosome–17 in five different resolutions of nomenclature as defined by ISCN [3].

essential ontological dichotomies of universals and particulars [6]. For example in cytogenetics, an application area of interest, International System for Human Cytogenetic Nomenclature (ISCN) has a standardized nomenclature for the parts of the genome. It has defined five different resolutions of the chromosome band: 300, 400, 550, 700, and 850 [3]. In other words, there are 862, and 311 regions in a genome in resolution 850 (fine resolution), and 300 (coarse resolution), respectively. Figure 1 shows an example of multiresolution data resulting from the ISCN nomenclature which is also our application area of interest. Figure 1 shows a part of chromosome–17 in five different resolutions forming a tree structure among different chromosome bands. Here, the same part of genome measured in different levels of detail generating multiresolution data.

Finite Mixture Models are semi-parametric probability density functions represented as the weighted sum of component densities of chosen probability distributions such as Gaussian, Bernoulli, or Poisson [7,8]. Mixture models have found wide spectrum of uses such as clustering [9], density estimation [10], modeling heterogeneity [11], handling missing data [12], and model averaging. They are versatile because of their suitability for any choice of data distribution, either discrete or continuous, and flexibility in the choice of component distributions [8]. However, mixture models in their basic form only operate on single data resolution, and is unable to model multiresolution data. The only mixture modeling solution to multiresolution data are to model the different resolutions separately and at best compare the findings. Cancer is not a single disease but a heterogeneous collection of several diseases [13]. Therefore, we use mixture models to model cancer patients discussed in Section 4.2 because mixture models are well known for their ability to model heterogeneity.

In our previous work, we transform the multiresolution data to a single resolution using different deterministic transformation methods, and model the resulting single resolution data [14]. Results in [14] shows improvement in the performance of mixture models through multiresolution analysis compared to single resolution analysis. We also proposed a multiresolution mixture model based on merging of mixture components across different resolutions in [15]. The improvement in [15] is that the models assimilate the information contained in other data resolutions. Furthermore, transformations here are in the model domain unlike in the data domain as in [14]. In all scenarios above, the mixture models are generated in a single resolution and directly unusable in multiresolution scenarios without modification.

In the past, research has considered formulating the multiresolution mixture model in different application areas. For instance, the multiresolution Gaussian mixture model in [16] approximates the probability density, and adapts to smooth motions. The design of the model is for a specific choice of data distribution, and generates trees of decreasing variance, and consequently a tree of Gaussians. Unlike an actual multiresolution model, this essentially models single resolution data using a tree from the same data for multiple Gaussians on different scales with different variance. Furthermore, difference in the pyramid structure present in other domains limit its general applicability.

Authors have increased the efficiency and robustness of learning mixture models using multiresolution kd-trees [9,17]. Furthermore, authors in [10] have proposed the a mixture of tree distributions in a maximum likelihood framework. Similarly, authors in [18,19] use multiresolution binary trees to learn discrete probability distribution. However, it is impossible to represent all multiresolution data as kd-trees which are binary in nature. In addition, the focus in [18,19] is in modeling single resolution data. Additionally, multiresolution trees are also used in object recognition [20], and as binary space portioning trees [21]. However, the authors in [20,21] use them in the context of recursive neural networks, and geometric representations for information visualization, respectively.

In this paper, we propose a multiresolution mixture model whose components are Bayesian networks denoting the hierarchical structure present in multiresolution data. We learn the parameters of each component distribution in a Bayesian network framework. Component distributions in the form of Bayesian network is useful also to impute the missing data resolutions considering them as missing values. Finally, we transform the multiresolution data in the Bayesian network representation to a single data in vector form to learn the mixing coefficients and the parameters of the mixture model in a maximum likelihood framework using the EM algorithm.

2 Bayesian Networks of Multiresolution Data

Bayesian networks bring the disciplines of graph theory and probability together to elegantly represent complex real-world phenomena dealing with uncertainty [22,23]. A Bayesian network consists of nodes or vertices that encode information about the system in the form of probability distributions, and links or arcs or edges that denote the interconnections or interactions between nodes in the form of conditional independence [22]. It analytically represents a joint distribution over a large number of variables. Furthermore, it treats learning and inference simultaneously, seamlessly merges unsupervised and supervised learning, and also provides efficient methods for handling missing data [23].

2.1 Component Distributions of Multiresolution Hierarchy as Bayesian Networks

A Bayesian network is a directed acyclic graph (DAG) that describes a joint distribution over the set of random variables $X_1, \ldots, X_d$ such that

$$P(X_1, \ldots, X_d) = \prod_{i=1}^{\mathrm{d}} P(X_i \mid \mathrm{parents}(X_i)), \tag{1}$$

where parents(X_i) are the set of vertices from which there is an edge to X_i. Figure 2 shows an example of a Bayesian network of six random variables A, B, C, D, E, and F. The six vertices represent the six random variables, and the five directed edges represent the conditional dependencies (independence). We can define conditional independence in a Bayesian network as: A variable is conditionally independent of all the variables in the network given its Markov blanket. The Markov blanket of a variable is the set of its parents, its children, and the other parents of its children.

Data in multiple resolutions share a commonality because they measure the same phenomenon. A single feature in the coarse resolution corresponds to one or more features in the fine resolution. We can exploit this information from the application area to determine the relationships between data resolutions

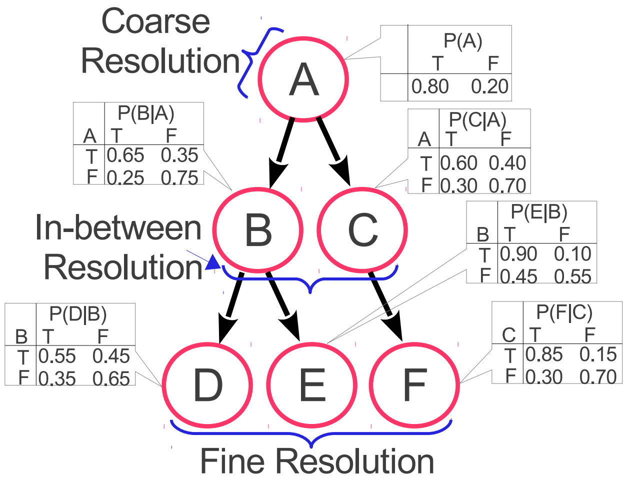


Fig. 2. Network representation of the multiresolution data where ancestors denote data in coarse resolution and leaves denote data in fine resolution. The network a simple Bayesian network of six random variables with five edges along with the associated probability tables.

and consequently, the structure of the Bayesian network. The data features in the coarse resolution form the root and branches near the root of the network. Similarly, the data features in the fine resolution form the branches towards the leaves and the leaves of the tree. Additionally, we can assume that the directed arrows originate from the features in the coarse resolution for computational efficiency.

Each vertex of a Bayesian network bears a corresponding conditional probability distribution (CPD). The CPD specifies that a child takes a certain value with a probability depending the value of its parents [22]. In the figure, for example the variables D, and E are conditionally independent given B. We can simplify the joint probability distribution of A, B, C, D, E, and F using the conditional independences in Figure 2 as:

$$\begin{aligned} P(A,B,C,D,E,F) &= P(A|B,C,D,E,F)P(B|A,C,D,E,F)\ldots \\ &\quad P(C|A,B,D,E,F)P(D|A,B,C,E,F)\ldots \\ &\quad P(E|A,B,C,D,F)P(F|A,B,C,D,E) \\ &= P(A)P(B|A)P(C|A)P(D|B)P(E|B)P(F|C) \qquad (2) \end{aligned}$$

The CPD of a discrete variable is represented in a table as shown in Figure 2. It enumerates each possible set of values for the variable and its parents. Algorithms based on maximum likelihood (MLE) and maximum a posteriori estimates (MAP) can learn the parameters of the Bayesian network with a known structure [24]. In our application, the structure of Bayesian networks comes from the domain knowledge. The depth of the Bayesian network depends on the number of resolutions in multiresolution data. Learning a Bayesian network of known structure involves determining the CPD of the variables in the network. We learn the CPD of the variables using the Maximum Likelihood Estimate (MLE) [22].

2.2 Missing Resolutions in Multiresolution Data

Missing data has received considerable research interest because of their abundant occurrence in many domains [12,25]. The problem of missing data escalates when some resolutions (entire data) in a multiresolution setting are missing. Therefore, when the values are missing in multiresolution analysis, one or more resolutions (entire data) will be missing. This is unlike the typical missing data problems where small number of variables in some samples will be missing. For example, data in a coarse resolution can be missing while data in other resolutions are available. Bayesian networks have a seamless ability to handle missing data [12]. Therefore, learning the Bayesian networks also helps to generate the data in missing resolutions because Bayesian networks are generative models.

We can impute the missing values using marginal inference in Bayesian Networks. Marginal inference is the process of computing the distribution of a subset of variables conditioned on another subset [22]. We can calculate the marginal inference for a joint distribution P(A,B,C) given the evidence B=true as:

$P(A \mid B = true) \propto \sum_{C} P(A, B = true, C).$

Authors have proposed algorithms such as variable elimination, and sum product algorithm to compute marginal inference [22]. We draw samples under the given evidence from consistent junction trees using the BRMLToolbox [22].

3 Multiresolution Mixture Model of Multivariate Bernoulli Distributions

Mixture Models are semi–parametric latent variable models that models a statistical distribution by a weighted sum of parametric distributions [7,8,22]. They are flexible for accurately fitting any complex data distribution for a suitable

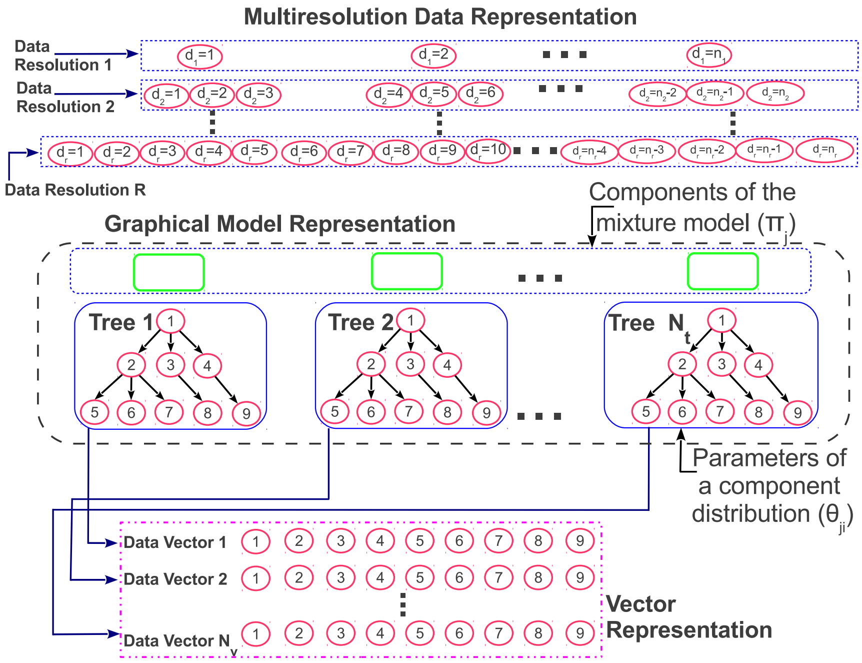

Multiresolution Data Representation
Data Resolution 1
Data Resolution 2
Data Resolution R
Graphical Model Representation
Components of the mixture model (π_j)
Tree 1
Tree 2
Tree N_t
Parameters of a component distribution (θ_ji)
Data Vector 1
Data Vector 2
Data Vector N_v
Vector Representation

Fig. 3. Top panel shows the general representation of multiresolution data. There are 'r' data resolutions of different dimensionality. The multiresolution data representation is then transformed to a Bayesian network representation which in turn is mapped to a vector of single resolution data. The Bayesian network representation in the middle panel (within the dashed rectangle) also depicts a mixture model having the Bayesian networks as the components for data in multiple resolutions. The three solid rectangles on the top represent different mixture coefficients. Similarly, the three network of nodes denote the three component distributions where each vertex defines a parameter of the component distribution. The numbers inside the nodes denote the position of the variable in the vector representation with regards to dimensionality. The dash–dotted rectangle in the bottom of the figure shows the vector representation of the data derived from the Bayesian network representation. In the figure, N_t, and N_v denote the number of networks abreast adjacently in the multiresolution data, and the number of samples in multiresolution data mapped to a vector representation, respectively.

choice of data distribution, and a good enough number of mixture components. Expectation Maximization (EM) algorithm helps to learn the maximum likelihood parameters of the mixture model [26]. The EM algorithm requires prior knowledge of the number of components in the mixture model. Model selection is the process of determining the number of components in the mixture model [8]. In our previous work, we have tried to solve the problem of model selection in mixture models [11,27,28]. We learn mixture model of different complexities in a cross-validation setting and select a model with the number of components that gives the best generalization performance.

A multiresolution data is a collection of different component distributions as shown in the middle panel (within the dashed rectangle) of Figure 3. The multiresolution components in the middle panel (within the dashed rectangle) of Figure 3 encode the relationships between different resolutions of multiresolution data. The structure of the component distribution comes from the domain knowledge. Thus, the problem with regard to model selection in multiresolution data culminates to determining the optimal number of such component distributions present in the data. Similarly, learning the parameters of the component distributions involves learning the parameters of those networks.

In the general framework for the EM algorithm, we can assign only a single probability value to a node in the mixture model [26]. However, each variable in Bayesian network consists of minimum of two probability values denoting the CPD of the nodes. Therefore, in this contribution, we map the Bayesian network to vector representation to learn a multiresolution mixture model of Bayesian networks. This simple and intuitive solution proposed in this contribution transforms Bayesian networks to vectors with increasing dimensionality representing increasing depth of the Bayesian network. The first element of the vector will be the root node in the first generation i.e. coarsest resolution. Similarly, the last element of the vector will be leaf node of the last generation i.e. finest resolution arranged from left to right as shown in the bottom panel (within the dashed dotted rectangle) of Figure 3. In the middle and the bottom panel of Figure 3, the number inside the vertices denote the relative position of the variable in the vector representation with regards to dimensionality. Multiresolution mixture components transformed to a vector representation will have the same dimensionality because the structure of component distributions are identical with one another. However, component distributions have different parameters.

Vector representation of Bayesian networks eases modeling multiresolution data in one resolution. Furthermore, it increases the number of data samples. The number of samples in the vector form, N_v will be $N_v = N \times N_t$. Here, N is the number of samples of data in each resolution. Similarly, N_t is the number of Bayesian networks present in the data along the dimension corresponding to one sample. Furthermore, the data dimensionality will be considerably reduced as the depth of the Bayesian network is generally small. Additionally, Bayesian networks provide the sparsity [24], making data dimensionality in the vector representation d_v is smaller than that of the finest resolution of the multiresolution data. Furthermore, the vector representation has larger number of data samples than the original Bayesian networks representation, $d_v < \max(d_r) \ll N_t \ll N_v$. Here, d_r is the dimensionality of data in different resolutions. An increase in the number of data samples and a reduction in dimensionality facilitates the learning of mixture models because they require a large number of samples to accommodate the increasing dimensionality of the data.

We can describe the mixture model of multivariate Bernoulli distributions for a 0-1 data [7] as:

$$p(\mathbf{x} \mid \Theta) = \sum_{j=1}^{\mathrm{J}} \pi_j \prod_{i=1}^{\mathrm{d}} \theta_{ji}^{x_i} (1 - \theta_{ji})^{1-x_i}. \tag{3}$$

Here, $\boldsymbol{\Theta} = \{J,\ \{\pi_j, \theta_j\}_{j=1}^{J}\}$ denotes the parameters of the mixture model of multivariate Bernoulli distributions. Here, π_j denotes the mixing proportions which sum to 1. Similarly, θ_{ji} is the probability that a random variable of the j^{th} component in the i^{th} dimension will take the value of 1. In multiresolution scenario, i differs for each resolution. Therefore, we have to model different resolutions with different models. We can formulate Equation 3 with respect to log likelihood to learn the mixture model using the EM algorithm in a maximum likelihood framework [8] as:

$$\mathcal{L}(\Theta) = \sum_{n=1}^{\mathrm{N}} \log\ P(x_n \mid \Theta) = \sum_{n=1}^{\mathrm{N}} \log \left[\sum_{j=1}^{\mathrm{J}} \pi_j \prod_{i=1}^{\mathrm{d}} \theta_{ji}^{x_{ni}} (1 - \theta_{ji})^{1-x_{ni}} \right]. \tag{4}$$

Given the number of mixture components, J, parameterized by $\Theta = \{\pi_j, \theta_j\}$, the EM algorithm can learn the mixture model that maximizes the likelihood in Equation 4. Model selection using cross-validated log likelihood can determine the number of mixture components, J [11,27,28].

4 Experimental Data

We experiment with the proposed methodology on two multiresolution data: an artificial data, and a chromosomal aberrations data, both in three resolutions.

4.1 Artificial Multiresolution 0-1 Dataset

In some application areas the relationships between different resolutions in a multiresolution setting are well known [6]. We can exploit such knowledge to artificially generate realistic multiresolution data. We initially fixed the structure of a Bayesian network to the the components of the mixture model shown in the middle panel (within the dashed rectangle) of Figure 3. Five such Bayesian networks were abreast along the dimensionality of the data. The dimensionality of data in three different resolutions are 5, 15, and 25 respectively. Firstly, we generate the data in the finest resolution i.e. having dimensionality of 25.

We fix two parameters to sample the data of given dimensionality: X is uniformly distributed in the range $[0,1]$, and l is normally distributed with mean 0.1 and standard deviation 0.04.

$$X \sim U\,[0,1] \times d \text{ and } l \sim N\left(\mu = 0.1, \sigma^2 = 0.04\right) \times d$$

where d denotes the data dimension.

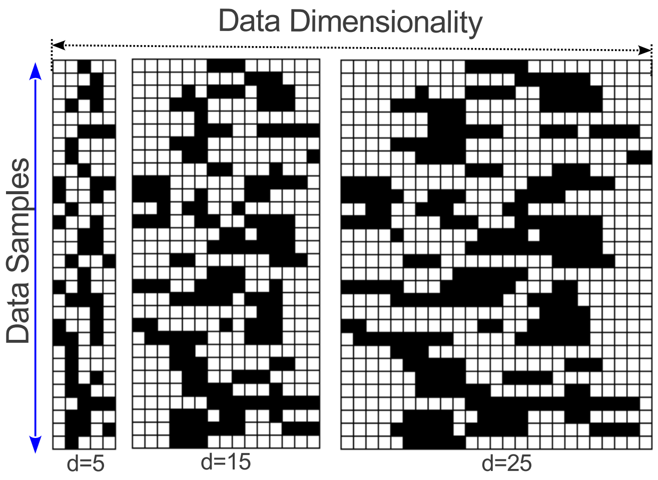


Fig. 4. First thirty samples of three different resolutions of the artificial 0-1 data. Black denotes 1s and white denotes 0s.

First, we create a matrix of the required size with all zeros. We divide the unit interval in 25 equal parts to generate 25–dimensional data. The parameter, X, defines the beginning of an aberration, and l defines the length of the aberration. We randomly choose a data sample for aberration and flip zeros to ones from dimensionality X of length l i.e. from dimension X to dimension $X+l$. We ignore the the lengths that are greater than the dimension $25 < (X+l)$ to maintain the data dimensionality. We continue this process iteratively until the number of 1s is approximately 55-60% of the data to mimic chromosomal aberration datasets.

Domain knowledge of chromosomal aberrations informs us about the typical length of aberrations. Furthermore, aberrations never span across the centromere. Therefore, we break an aberration that is longer than a predefined length, 15 in the experiments, randomly either on the left or the right of the centromere. We fix the centromere after the 10^{th} dimension, i.e. variable on the 10^{th} dimension is on the left side of the centromere, and the variable on the 11^{th} dimension is on the right side of the centromere.

Figure 4 shows artificial data in three different resolutions with dimensionalities 5, 15, and 25, respectively. Figure 4 also shows that similar to chromosomal aberration data, the artificial data are sparse, and spatially dependent. We gain the knowledge of relationships between data in different resolutions from the

Bayesian network as shown in the middle panel (within the dotted rectangle) of Figure 3. We apply that knowledge to downsample the data in dimensionality 25 to a dimensionality of 15, and then 5 using the majority voting downsampling method proposed in [14]. In experiments we fix the number of samples of the dataset to 1000 which is similar to the chromosomal aberration dataset.

4.2 Multiresolution Chromosomal Aberration Dataset

The causes and consequences of chromosomal aberration such as amplification, deletion, and duplication have significant roles in cancer research [13]. DNA copy number amplifications have been defined as the hallmarks of cancer [11,28]. Chromosomal aberrations are early markers of cancer risk and their detection and analysis has considerable clinical merits [11,29]. The two chromosomal aberration datasets in two resolutions have a dimensionality of 393, and 862 in coarse, and fine resolution, respectively. Both datasets are used in [11,27,28]. Datasets are available from the authors on request. The sources of the two datasets were different and correspond to different cancer patients in the two different resolutions.

We experiment chromosome–wise to constrain the complexity of learning the mixture model because the data are high dimensional and samples are small. Complexity of mixture models increases quadratically with the dimensionality. For example, the number of samples in chromosome 17 is 342 and 2716 in coarse and fine resolution, respectively. Therefore, we correspond the first 342 samples in fine resolution to the samples in coarse resolution. We then downsample the next 342 samples (samples 343 to 684) to a resolution between coarse and fine resolution such that the depth and structure of the resulting Bayesian network are similar to those of the networks used for artificial dataset. We ignore the remaining 2032 samples in the fine resolution.Similarly, the network present in the real world dataset differ from the artificial dataset. We select the most representative network covering more than 50% of the data and ignore the other networks. The structures of the networks are similar (often number of types of trees in the dataset is about ≈ 3).

5 Experiments and Results

The experimental studies in this paper are a two-step procedure because the algorithm models multiresolution data in two steps. Firstly, we learn the component distributions in a Bayesian network framework from different resolutions of the data. Secondly, we model multiresolution data after transforming the Bayesian networks to vectors using mixture models.

5.1 Experiments with Bayesian Networks of Multiresolution Data

From the knowledge of multiresolution data relationships in Section 4, we generate the five different Bayesian networks abreast adjacently along the dimensionality of the data. We use BRMLToolbox to encode and generate Bayesian network [22]. We use a maximum likelihood framework to learn the conditional probabilities of the network.

As discussed in Section 2.2, some of the resolutions (entire datasets) can be missing in a multiresolution scenario. We use the prowess of Bayesian networks in handling missing data [12] to impute missing resolutions. In experimental setup: firstly, we learn the parameters of the component distributions in Bayesian network framework via maximum likelihood. We then ascertain the performance of component distributions as Bayesian network models especially with respect to their ability to impute missing values. We artificially generate two scenarios: where one, and two resolutions of data are missing. We draw the samples from a consistent junction tree in the Bayesian network under the given evidence using BRMLToolbox [22]. The number of samples equal that of the original dataset. We then compare the re-sampled data with the original data.

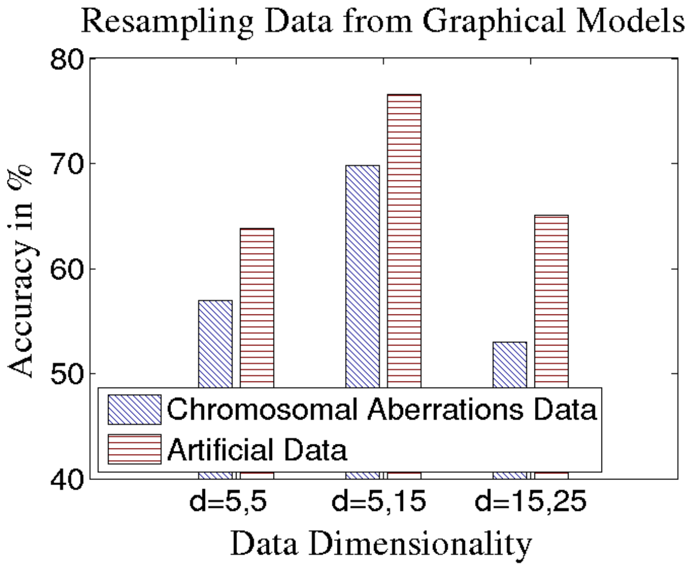


Fig. 5. The accuracy in re-sampling the data in missing resolutions conditioned on the data in other available resolutions. Comma in the X-axis separates the dimensionalities of the artificial data and the chromosomal aberration data, i.e. d=x,y denotes dimensionality (d) = (chromosomal aberrations data dimension (x), artificial data dimension (y)).

We calculate the matrix difference between the original data and the data sampled from the Bayesian networks. The difference in the binary data is sum of the number of places where 0s are 1s, and 1s are 0s. The difference is comparatively small in smaller dimensions than in the larger dimensions because the cumulative difference depends on the size of the dataset. Therefore, we calculated the accuracy of element-wise matching of two datasets as shown in Figure 5. Accuracy is the percentage of places where each element of both the matrices are equal.

In artificial data, when the data resolution with dimensionality 15 is missing, other data resolutions with dimensionality 5, and 25 are available, and we need to impute only data in resolution 15. Similarly, Data resolutions with dimensionalities 5, and 25 are missing when the data resolution with dimensionality 15 is available. Therefore, we should impute both the resolutions with dimensionalities 5, and 15. In this case, we can simply run single resolution analysis. However, we try to artificially create this scenario to demonstrate that our algorithm performs reliably under harsh conditions of large amount of missing values. In chromosomal aberrations data, coarse and the in–between data resolutions have the same dimensionality of 5. Therefore, when the in–between data resolution with dimensionality 5 is missing, coarse, and fine data resolution with dimensionality 5, and 15 are available, respectively. Similarly, the coarse, and fine data resolutions with dimensionalities 5, and 15 are missing when the data in the in–between resolution with dimensionality 5 are available.

The results in Figure 5 show that accuracy of matching is higher when two resolutions of data are available and only the data in a single resolution are missing. When only one data resolution is available, and we need to impute two resolutions in the coarse and the fine resolution, the accuracy is poorer. This result is intuitive because the number of known variables is smaller than the number of missing variables when two data resolutions are missing. Similarly, accuracy is poorer in high dimensional data (fine resolution) compared to data with lower dimensionality (coarse resolution). This discrepancy is the result of the curse of dimensionality phenomenon. Overall, the results show that the model of component distributions as Bayesian networks produces plausible results.

5.2 Experiments on Mixture Modeling of Multiresolution Data

In experimental setup, firstly, we transform the multiresolution data to the Bayesian network representation as shown in the Figure 3. Secondly, we transform the Bayesian network representation to the vector representation after imputing missing values (if any) as explained in Section 3. In the vector representation, the transformed multiresolution data have same dimensionality. The EM algorithm learns the mixture model with a priori knowledge of the number of components for data. As in [11,27,28], we use model selection in a 10-fold cross-validation setting to select the appropriate number of mixture components.

We train models of different complexities in a ten-fold cross–validation setting and select the model with the best generalization performance. Figure 6 shows that both training and validation likelihood steadily increases until the number of components is 5, then smoothen and flatten after the number of components is 5. This suggests that 5 is the appropriate number of mixture components.

After selecting the number of components, we train 200 different models of the same complexity and choose the model that produces the best likelihood on the data to ameliorate the problem of local optima in the EM algorithm. We also perform similar experiments with data in each resolution to select the number of mixture components and train the mixture model as a comparison with the results of the multiresolution model. Table 1 shows the variation in the number

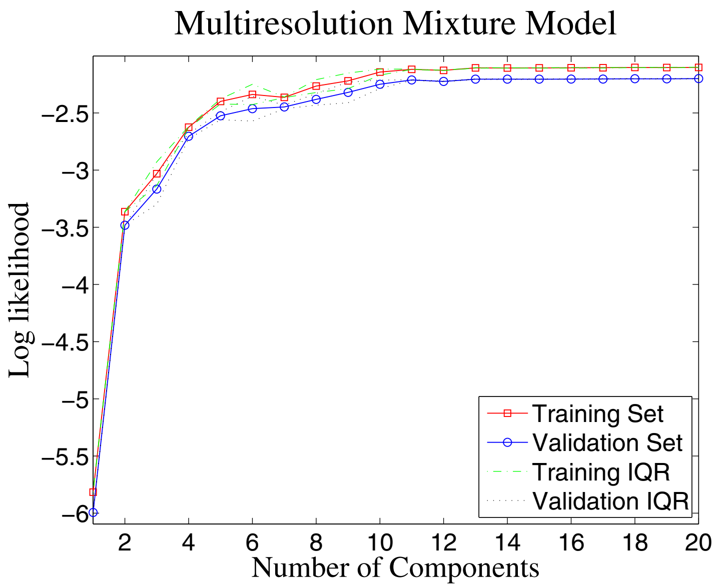


Fig. 6. Model selection in a 10-fold cross-validation setting in multiresolution artificial data. Final results for the same model selection are denoted by a boldfaced row in Table 1. Averaged training and validation likelihood along with their corresponding Inter Quartile Range (IQR) for each training and validation run has also been plotted. The selected number of components is 5.

Table 1. The results of mixture modeling on single resolution and multiresolution models. Here, J and $\mathcal{L}$, denote the selected number of components and the log likelihood obtained by the best model, respectively.

Artificial Data			**Chromosome-17**		
Datasets (Dimension)	**Results**		**Datasets Dimension)**	**Results**	
	J	$\mathcal{L}$		**J**	$\mathcal{L}$
Single Resolution (5)	3	-3.24	Single Resolution (5)	4	-2.23
Single Resolution (15)	6	-8.32	Single Resolution (5)	3	-2.17
Single Resolution (25)	7	-12.84	Single Resolution (15)	5	-3.73
Multiresolution (9)	**5**	**-2.40**	Multiresolution (5)	4	-2.14

of components required to fit the data in different resolutions. Furthermore, the likelihood is considerably smaller in single resolution showing improvement in mixture modeling because of the use of multiple resolutions.

Figure 7 shows the log likelihood of three single resolution models and a multiresolution model. We trained the mixture model initialized at random in a ten-fold cross-validation setting with the selected number of components to convergence, i.e. until the increase in log-likelihood is small, 0.0001 in the experiments. The shorter the bar the better the result as Y-axis depicts negative log likelihood.

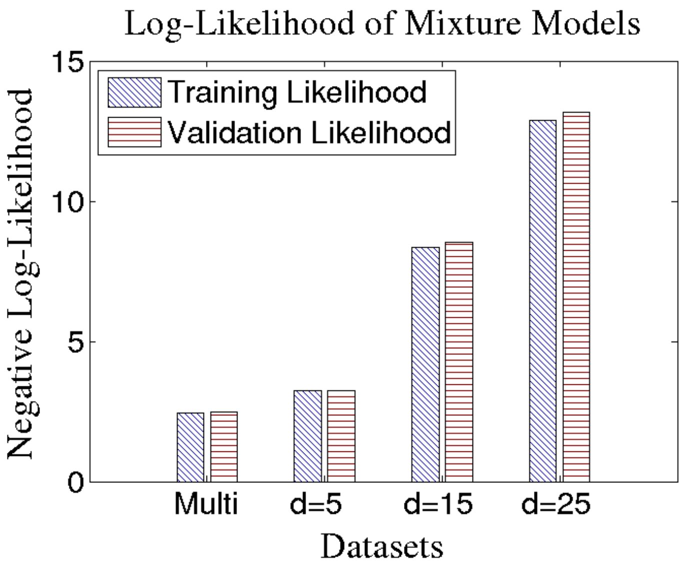


Fig. 7. The log likelihood of three mixture models in single resolution and a multiresolution mixture model trained in a 10-fold cross-validation setting after selecting the number of components. The Y-axis shows the negative log likelihood, therefore, the shorter the bar, the better the result.

We select a different number of components for each dataset as shown in Table 1. The results also show that multiresolution mixture model outperforms the single resolution models. Log likelihood is comparatively smaller in dimensionalities of 15, and 25 because of the increased dimensionality of the data. The likelihood of the multiresolution model is better than the data with the smallest dimensionality of 5 in single resolution although the dimensionality of the multiresolution data is 9. The Table 1 also shows similar results on chromosomal data.

6 Summary and Conclusions

In this paper, we proposed a mixture model of multiresolution components to model multiresolution 0-1 data. Firstly, we design the multiresolution components of the mixture model as Bayesian networks with the knowledge of the hierarchy of resolutions from the domain ontology. We then learn the CPD of the networks from the multiresolution data. Secondly, we transform the multiresolution component distributions to vector representation and learn the mixture model in a ten-fold cross validation setting. We experimented with the algorithm on a multiresolution artificial dataset and also on a multiresolution chromosomal aberration dataset. The experimental results show that the proposed approach of multiresolution modeling outperforms single resolution models.

Acknowledgments. The work is funded by Helsinki Doctoral Programme in Computer Science – Advanced Computing and Intelligent Systems (**Hecse**), and Finnish Centre of Excellence for Algorithmic Data Analysis Research (**ALGODAN**). I would also like to thank colleague Mikko Korpela for reading through the manuscript and suggesting the improvements in presentation.

References

1. Garland, M.: Multiresolution Modeling: Survey & Future Opportunities. In: Eurographics 1999 – State of the Art Reports, pp. 111–131 (1999)
2. Willsky, A.S.: Multiresolution Markov Models for Signal and Image Processing. Proceedings of the IEEE 90(8), 1396–1458 (2002)
3. Shaffer, L.G., Tommerup, N.: ISCN 2005: An International System for Human Cytogenetic Nomenclature(2005) Recommendations of the International Standing Committee on Human Cytogenetic Nomenclature. Karger (2005)
4. Lindeberg, T.: Scale-space theory: A basic tool for analysing structures at different scales. Journal of Applied Statistics 21(2), 224–270 (1994)
5. Vetterli, M., Kovačevic, J.: Wavelets and Subband Coding. Prentice-Hall, Inc., Upper Saddle River (1995)
6. Russell, B.: On the Relations of Universals and Particulars. Proceedings of the Aristotelian Society 12, 1–24 (1911)
7. Everitt, B.S., Hand, D.J.: Finite Mixture Distributions. Chapman and Hall, London (1981)
8. McLachlan, G.J., Peel, D.: Finite Mixture Models. Probability and Statistics – Applied Probability and Statistics Section, vol. 299. Wiley, New York (2000)
9. Moore, A.: Very Fast EM-based Mixture Model Clustering Using Multiresolution KD–trees. In: Kearns, M., Cohn, D. (eds.) Advances in Neural Information Processing Systems, pp. 543–549. Morgan Kaufmann (April 1999)
10. Meilâ, M., Jordan, M.I.: Learning with mixtures of trees. Journal of Machine Learning Research 1, 1–48 (2000)
11. Myllykangas, S., Tikka, J., Böhling, T., Knuutila, S., Hollmén, J.: Classification of human cancers based on DNA copy number amplification modeling. BMC Medical Genomics 1(15) (May 2008)
12. Marlin, B.M.: Missing data problems in machine learning. PhD thesis, University of Toronto (2008)
13. Kirsch, I.R.: The Causes and Consequences of Chromosomal Aberrations, 1st edn. CRC Press (December 1992)
14. Adhikari, P.R., Hollmén, J.: Patterns from multiresolution 0-1 data. In: Proceedings of the ACM SIGKDD Workshop on Useful Patterns, UP 2010, pp. 8–16. ACM, New York (2010)
15. Adhikari, P.R., Hollmén, J.: Multiresolution Mixture Modeling using Merging of Mixture Components. In: Hoi, S.C.H., Buntine, W. (eds.) Proceedings of the Fourth Asian Conference on Machine Learning, ACML 2012, JMLR Workshop and Conference Proceedings, Singapore, vol. 25, pp. 17–32 (2012)
16. Wilson, R.: MGMM: multiresolution Gaussian mixture models for computer vision. In: Proceedings of 15th International Conference on Pattern Recognition, vol. 1, pp. 212–215 (2000)

17. Ng, S.-K., McLachlan, G.J.: Robust Estimation in Gaussian Mixtures Using Multiresolution Kd-trees. In: Sun, C., Talbot, H., Ourselin, S., Adriaansen, T. (eds.) Proceedings of the 7th International Conference on Digital Image Computing: Techniques and Applications, pp. 145–154. CSIRO Publishing (2003)
18. Bellot, D.: Approximate discrete probability distribution representation using a multi–resolution binary tree. In: Proceedings of 15th IEEE International Conference on Tools with Artificial Intelligence, pp. 498–503 (2003)
19. Sanchís, F.A., Aznar, F., Sempere, M., Pujol, M., Rizo, R.: Learning Discrete Probability Distributions with a Multi-resolution Binary Tree. In: Corchado, E., Yin, H., Botti, V., Fyfe, C. (eds.) IDEAL 2006. LNCS, vol. 4224, pp. 472–479. Springer, Heidelberg (2006)
20. Bianchini, M., Maggini, M., Sarti, L.: Object Recognition Using Multiresolution Trees. In: Yeung, D.-Y., Kwok, J.T., Fred, A., Roli, F., de Ridder, D. (eds.) SSPR&SPR 2006. LNCS, vol. 4109, pp. 331–339. Springer, Heidelberg (2006)
21. Huerta, J., Chover, M., Quiros, R., Vivo, R., Ribelles, J.: Binary space partitioning trees: a multiresolution approach. In: Proceedings of 1997 IEEE Conference on Information Visualization, pp. 148–154 (1997)
22. Barber, D.: Bayesian Reasoning and Machine Learning. Cambridge University Press (2012)
23. Jordan, M.I.: Graphical Models. Statistical Science (2004)
24. Heckerman, D.: A Tutorial on Learning With Bayesian Networks. In: Jordan, M.I. (ed.) Learning in Graphical Models, pp. 301–354. MIT Press, USA (1999)
25. Enders, C.K.: Applied Missing Data Analysis, 1st edn. The Guilford Press (2010)
26. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B 39(1), 1–38 (1977)
27. Adhikari, P.R., Hollmén, J.: Fast Progressive Training of Mixture Models for Model Selection. In: Ganascia, J.-G., Lenca, P., Petit, J.-M. (eds.) DS 2012. LNCS (LNAI), vol. 7569, pp. 194–208. Springer, Heidelberg (2012)
28. Tikka, J., Hollmén, J., Myllykangas, S.: Mixture Modeling of DNA copy number amplification patterns in cancer. In: Sandoval, F., Prieto, A.G., Cabestany, J., Graña, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 972–979. Springer, Heidelberg (2007)
29. Lu, X., Shaw, C.A., Patel, A., Li, J., Cooper, M.L., Wells, W.R., Sullivan, C.M., Sahoo, T., Yatsenko, S.A., Bacino, C.A., Stankiewicz, P., Ou, Z., Chinault, A.C., Beaudet, A.L., Lupski, J.R., Cheung, S.W., Ward, P.A.: Clinical Implementation of Chromosomal Microarray Analysis: Summary of 2513 Postnatal Cases. PLoS ONE 2(3), e327 (2007)

Model Tree Ensembles for Modeling Dynamic Systems

Darko Aleksovski[1,3], Juš Kocijan[1,2], and Sašo Džeroski[1,3]

[1] Jožef Stefan Institute, Jamova 39, Ljubljana, Slovenia
[2] University of Nova Gorica, Vipavska 13, Nova Gorica, Slovenia
[3] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
{darko.aleksovski,jus.kocijan,saso.dzeroski}@ijs.si

Abstract. We address the task of discrete-time modeling of nonlinear dynamic systems using measured data. In the area of control engineering, this task is typically converted into a classical regression problem, which can then be solved with any nonlinear regression approach. As tree ensembles are a very successful predictive modelling approach, we investigate the use of tree ensembles for regression for this task.

While ensembles of regression trees have been extensively used and different variants thereof explored (such as bagging and random forests), ensembles of model trees have received much less attention, being limited mostly to bagging of model trees. We introduce a novel model tree ensemble approach to regression, namely, bagging and random forests of fuzzified model trees. The main advantage of the new approach is that it produces a model with no discontinuities with a satisfactory extrapolation behavior, needed for modeling dynamic systems.

We evaluate existing tree ensemble approaches to regression and the approach we propose on two synthetic and one real task of modeling nonlinear dynamic systems coming from the area of control engineering. The results show that our proposed model tree ensembles outperform ensembles of regression trees and have comparable performance to state-of-the-art methods for system identification typically used in control engineering. The computing time of our approach is comparable to that of the state-of-the-art methods on the small problems considered, with the potential to scale much better to large modeling problems.

1 Introduction

Dynamic systems are systems that change over time. The task of modeling dynamic systems is of high practical importance, because such systems are ubiquitous across all areas of life. The models allow for better understanding of dynamic systems, as well as their control, the latter being the focus of study in control engineering.

In control engineering, a dynamic system is typically represented by two (sets of) variables. System (endogenous) variables, denoted by y, describe the state that the system is in at any particular point in time. Input (exogenous) variables, denoted by u, capture external conditions that influence, i.e., are relevant to, the system.

Dynamic systems can be modeled in continuous time with systems of ordinary differential equations, describing the rate of change for each of the system variables. They can also be modeled in discrete time, by using difference equations that describe the state of the system at (a discrete) time point k as a function of previous system states and inputs. The task of constructing models of dynamic systems, both in continuous and discrete time, from measured data is the topic of study of the system identification sub-area of control engineering.

The Task of Discrete-Time Modeling of Dynamic Systems

This paper addresses the task of discrete-time modeling of nonlinear dynamic systems. As mentioned above, two types of variables are used in modeling, system (endogenous) and input (exogenous) variables, denoted by y and u, respectively. Using the external dynamics approach [1], the task of empirical modeling of a dynamic system can be formulated as a regression problem of finding a difference equation that fits an observed behavior of the system.

More precisely, to model a system described by y and u, we need to formulate a difference equation that expresses the value of the state variable y at a given time point k as a function of past system and input variables (y and u).

The transformation creates a new vector of features which is composed from the lagged values of the input variable u and system variable y. Typically, up to n time points in the immediate past (with respect to k) are considered.

At time point k, the dynamic system is thus represented by the vector of features $\mathbf{x}(k)$

$$\mathbf{x}(k) = [u(k-1), u(k-2), .., u(k-n), y(k-1), y(k-2), .., y(k-n)]^T \qquad (1)$$

where n is the dynamic order (lag) of the system. The model of the system is a difference equation that describes the state of the system at (a discrete) time point k, $y(k)$ as a function of the previous system states and inputs (i.e., $\mathbf{x}(k)$). The corresponding regression problem is to train a nonlinear function approximator $f(.)$, s.t. $y(k) = f(\mathbf{x}(k))$, from a table of data generated from an observed behavior in the manner described above.

Early research in discrete-time system identification focused on parametric and semi-parametric regression approaches. More recently, non-parametric approaches dominate. These include the widely used basis-function approaches of Artificial Neural Networks [1] and fuzzy modeling, as well as kernel methods [2] and Gaussian Process models [3].

In this context, we propose the use of another non-parametric approach to regression, coming from the area of machine learning. Tree ensembles are very well known and perform well in terms of predictive power. They are also very efficient and scale well to large problems. We thus set out to investigate the use of tree ensembles for regression for the above task.

To our knowledge, the present work is the first to consider the use of tree ensemble methods for modeling dynamic systems.

Tree Ensembles for Regression

The simplest ensemble creation procedures is based on the resampling principle and is known as Bootstrap Aggregation (Bagging) [4]. It generates ensembles from randomly sampled bootstrap replicates (random samples with replacements) of the training set. Each bootstrap replicate is used to build a base model, using the base learner algorithm. Hopefully, each base model would capture and represent a different hypothesis and the combination of the base models would be a more accurate predictor than each base model. Bagging can be seen as an ensemble method that modifies the training data.

Later work by Breiman [5] concluded that ensembles could benefit from modifying the learning process as well. This tree ensemble methodology is known as Random Forests and includes randomization of the attributes considered for split nodes of the trees. Both tree ensemble approaches use a regression tree learning algorithm as a base learner, which only provides a piecewise constant approximation of the true function.

However, the modeling of nonlinear dynamic systems using the transformation to a regression task requires approximating (smooth) nonlinear functions. Also, the extrapolation behavior of the model is important because the measured (training) data for the dynamic system cannot capture all parts of the operating region being modeled [6]. The existing piecewise constant fit that a regression tree or an ensemble of regression trees provide is not sufficient for this task. An obvious solution of extending the ensembles of regression trees methodology is introducing ensembles of model trees, which learn linear models for the terminal nodes.

Linear model trees provide a piecewise linear approximation and are a more powerful approximator which also handles the extrapolation problem better than regression trees. Another approach that is often pursued in the system identification community, for modeling dynamic systems, is utilizing fuzzy methods. In the context of model trees, fuzzy approaches are able to provide continuous transitions between neighboring local (linear) models and a more accurate fit to the true nonlinear function.

While ensembles of regression trees have been used extensively and different variants thereof have been explored [7,8,5], ensembles of model trees have received much less attention in the machine learning literature. In fact, they have been mostly limited to bagging of model trees. Here we introduce a novel model tree ensemble approach to regression, namely, bagging and random forests of fuzzified model trees, which produce models with no discontinuities and with satisfactory extrapolation behavior.

Paper Outline

In this paper, we introduce a novel model tree ensemble approach to regression, namely, (bagging and) random forests of fuzzified model trees (Section 2). The main advantage of the new approach is that it produces a model with no discontinuities with a satisfactory extrapolation behavior. As explained above, this is needed for modeling dynamic systems.

We then evaluate existing tree ensemble approaches to regression and the approach we propose on two synthetic and one real task of modeling nonlinear dynamic systems coming from the area of control engineering. We also consider two state-of-the-art approaches to system identification coming from that area. Section 3 describes the experimental setup, while Section 4 presents the results.

The results show that our proposed model tree ensembles outperform ensembles of regression trees. They have comparable performance to the two state-of-the-art methods for system identification that are typically used in control engineering. The computing time of our approach is comparable to that of the state-of-the-art methods on the small problems considered, with the potential to scale much better to large modeling problems.

2 Model Trees Ensembles

The Model Trees Ensembles (MTE) methodology works by learning fuzzified linear model trees from bootstrap samples of the training data. The pseudocode for the Model Tree Ensembles methodology is shown in Table 1. The base learner algorithm starts by learning crisp linear model trees, where an example follows exactly one branch of a split in the tree for tests of the form $[A < v]$ (v is a crisp threshold). The next stage involves a tree fuzzification procedure, which transforms each (crisp) split of the trees into a fuzzy split, where an example can be sorted down both branches resulting from a split and the two corresponding predictions are combined. The prediction of the model tree ensemble $E = \{T_1, T_2, .., T_m\}$ is a uniformly weighted average of the individual tree predictions:

$$\hat{f}(\boldsymbol{x}) = \frac{1}{m} \sum_{T \in \{T_1, T_2, .., T_m\}} \hat{f}_T(\boldsymbol{x}) \tag{2}$$

where $\hat{f}_T$ is the prediction of model tree T.

The crisp model tree learning is based on the well-known M5′ algorithm [9,10]. The randomized base learner for building fuzzified model trees operates in the following stages: tree growing phase, tree pruning phase, and tree fuzzification phase. The tree growing phase includes a randomization of the split attribute selection, in line with the ideas from Random Forests [5]. The different stages of the tree learning algorithm are described in the sections that follow, and the pseudocode is shown in Tables 2, 3 and 4.

2.1 Tree Growing Phase

Tree growing is a recursive procedure that generates the initial structure of the tree. The procedure consists of determining whether the tree node should be a split (inner) node or a terminal node containing a linear model. If a split node is created, the procedure continues recursively for the examples sorted down each of the two branches created by the split.

The decision to create a terminal node instead of a split node (i.e. to perform pre-pruning) is taken when one of the two stopping criteria are met. The first criterion tests if the number of training points in the current node is smaller than the value of the minimal number of instances parameter (t). The second criterion stops tree growing when the standard deviation of the target attribute on the data points falling in the current node is smaller than 5% of its standard deviation on the whole training set.

The selection of the split parameters (the feature attribute to split on and the cut-point) is guided by the standard deviation reduction (SDR) heuristic, shown in (3) below. Normally the split with the highest reduction in the standard deviation is chosen. However, a randomization of the split attribute selection process can also be implemented in the tree growing phase, as described below.

Randomization of Attribute Selection. We consider a randomization of the base learning algorithm as in the random forests approach: instead of considering all features, only a random subset of features is considered when selecting the best split in a given node. A different random subset of the same size is considered in different nodes. The size is determined by the attribute randomization parameter p. For each candidate attribute in the subset, the standard deviation reduction (SDR) heuristic is used to evaluate all possible split cut-points. The feature attribute (A) and cut-point (c) combination in the test $[A < c]$ which maximizes the SDR heuristic is selected and used as a split at the current tree node. The SDR heuristic score is calculated as:

$$SDR = \sigma_S^2 - \frac{|S_l|}{|S|}\sigma_{S_l}^2 - \frac{|S_r|}{|S|}\sigma_{S_r}^2 \tag{3}$$

where S is the set of data points falling in the current tree node, S_l and S_r are the two subsets of data points corresponding to the left and right branches of the split. σ_S^2 denotes the standard deviation of the target attribute in the set S.

2.2 Tree Pruning Phase

It is well known that overly large trees are prone to overfitting. Tree pruning (shown in Table 3) is a method that handles overfitting by removing tree nodes which may deteriorate the performance of the tree. As a first step, linear models are estimated in all nodes of the tree, by using linear regression. The estimation of linear models also includes an attribute removal part: Features whose effect is small are dropped from the linear model [9].

After the estimation of linear models, the bottom-up pruning function evaluates whether to prune each tree node. It compares the accuracy of the linear model learnt for a node to the accuracy of the subtree rooted at the node. A decision to prune (replace the subtree rooted at that node with a terminal node) is made only if the accuracy of the subtree is smaller than the accuracy of the linear model.

Our experience showed that including unpruned model trees in the ensemble resulted in deterioraion of the ensemble performance. The linear regression procedure for the terminal nodes that contained smaller number of data points resulted in invalid linear models. One possible solution to this problem was to include a tree pruning method, which increased the performance of the ensembles.

2.3 Fuzzification

In order to smooth out the discontinuous transitions between two neighboring linear models of a model tree (i.e. linear models placed in two nodes following the same split node), we implement a split fuzzification procedure. A crisp split of the form $s[A < c]$, where A is a descriptive attribute and c is a cut-point, is transformed to a fuzzy split: A data instance now belongs to both the left and right subtrees of the split with probabilities $\mu_{A<c}$ and $\mu_{A\geq c}$ respectively. The probabilities for membership to the left and right subtrees are calculated by using a sigmoidal membership function with a single parameter: α - the inverse split width, as defined by (4) below.

$$\mu_{A<c} = \frac{1}{1+\exp(-\alpha(A-c))}, \qquad \mu_{A\geq c} = 1-\mu_{A<c} \tag{4}$$

The value of the split parameter α is different for each split. It is calculated so that the overlap between the two neighboring partitions that the split creates in the dimension of the split variable A is equal to some percentage w of the range of values for that attribute. In the experiments, the value of the parameter w is optimized using cross-validation.

2.4 Implementation

The Model Tree Ensembles method is implemented in Java, using the WEKA data mining software [11]. The WEKA implementation of the M5$'$ algorithm [9] for building model trees has been modified to include the changes described here.

In the Model Tree Ensembles method we extend the M5$'$ algorithm, but with some of its features disabled. The disabled features consider the learning of the linear models and the smoothing procedure.

The linear models in a node of an M5$'$ tree are learnt only using variables found in tests in the subtree rooted at that node. Instead of keeping this feature we learn the linear models by using all available features. Also, we turn off the smoothing procedure of M5$'$. Our preliminary experiments have shown that there is no difference in performance between ensembles which include and those that do not include the two features of the base learner.

3 Experimental Setup

In this section, we describe the datasets used for the experimental evaluation, the methods that we compare, the optimization of the parameters and the metrics used for the evaluation of the results.

Table 1. Pseudocode for the top level of the Model Tree Ensembles method

```
**Learn_ensemble**($S$)
Input: $S$ - training set
Output: $E$ - an ensemble
Create $m$ bootstrap samples of $S : S_1, S_2, .., S_m$
For $k = 1 \to m$ *do*
    $T_k =$ **Build_tree**($S_k$)
    $T_k =$ **Prune**($T_k$)
Let $w_{opt} =$ **Optimize_overlap_width_parameter**($\{T_1, T_2, .., T_m\}, S$)
For $k = 1 \to m$ *do*
    $T_k =$ **Fuzzify**($T_k, w_{opt}$)
Return ensemble $E = \{T_1, T_2, .., T_m\}$
```

3.1 Datasets

We have used three dynamic system datasets from the area of control engineering, more precisely process-industries. One of the datasets is measured, while the other two are synthetic. Of the latter, one is noise-free and the other is noisy.

The first dataset concerns the modeling of a unit for separating gas from liquid, which is part of a larger pilot plant [12]. It is a dataset with measured values concerning a semi-industrial process plant. The dynamic system is represented by 4 variables: three input (state of first valve, state of second valve, level of liquid in tank) and one system (pressure of gas in tank) variable. The dynamic order (lag) selected for this dataset is 1, which means that the regression dataset contains 4 features and 1 target variable. The dataset is already split into a training and test part, each of which contains 733 data instances.

The second dataset is generated from a dynamic system model that concerns the control of pH neutralization [13]. This synthetic dataset contains one input variable and one system variable. The selected dynamic order (lag) is 2, which means that the regression dataset contains 4 features and 1 target variable. This dataset too is split into a training and a test part, each having 320 data points. The training and testing sets for both datasets are obtained from different signals generated under the same conditions.

The third dataset is generated from the same dynamic system model as the second one, but with added noise. In more detail, the noise is added only to the system variable and only in the training set. In the final regression dataset, this means that two of the four feature variables (the ones corresponding to the system variable of the dynamic system model), as well as the target variable, contain noise in the training set. The noise added is white noise with a standard deviation equal to 20% of the standard deviation of the target variable.

3.2 Methods and Parameter Settings

We compare MTEs with two selected methods typically used in system identification. The first method that we compare to is the method of Neural Networks

Table 2. Pseudocode for the tree growing phase of the Model Tree Ensembles method

```
Build_tree($S$)
Input: $S$ - a training set
Output: $T$ - a tree
If $|S| < t$
    Return a terminal node
If standard deviation pre-pruning criterion is satisfied
    Return a terminal node
Let $\{A_1, A_2, .., A_p\}$ be a random subset of feature attributes
Initialize $s_{best}$
For $k = 1 \rightarrow p$  do
    Let split $s^*[A_k < c] = \underset{s}{argmax}(SDR(s[A_k < c]))$
    If $SDR(s^*) < SDR(s_{best})$
        $s_{best} = s^*$
Split set $S$ into subsets $S_l$ and $S_r$ based on split $s_{best}$
Let $T_l$ = Build_tree($S_l$),    $T_r$ = Build_tree($S_r$)
Return a tree with a split node $s_{best}$ and subtrees $T_l$ and $T_r$
```

Table 3. Pseudocode for the tree pruning phase of the Model Tree Ensembles method

```
Prune($T$)
Input: $T$ - a tree
Output: pruned tree
If root of $T$ is a split node
    Prune( $T \rightarrow$ left )
    Prune( $T \rightarrow$ right )
    Learn a linear model for the root node of $T$
    Calculate the error of the linear model $err_{LM}$
    If Subtree_error($T$) $> err_{LM}$ then
        Convert root of $T$ to a terminal node
Return $T$

Subtree_error($T$)
Input: $T$ - a tree
Output: numeric value
If root of $T$ is a split node
    Let $T_l$=$T \rightarrow$ left,    $T_r$=$T \rightarrow$ right
    Let $S$=$T \rightarrow$ examples,    $S_l$=$T_l \rightarrow$ examples,    $S_r$=$T_r \rightarrow$ examples
    Return ( $|S_l|$ * Subtree_error($T_l$) + $|S_r|$ * Subtree_error($T_r$) ) / $|S|$
Otherwise
    Return the error of the linear model in root of $T$
```

Table 4. Pseudocode for the fuzzification phase of the Model Tree Ensembles method

```
Fuzzify($T, w$)
Input: $T$ - a tree, and $w$ - overlap width parameter
Output: fuzzified tree
If root of $T$ is a split node
    Let the split at root of $T$ be $s[A_k < c]$
    Let $[x_k^{min}, x_k^{max}]$ be the range of the data points in dimension $k$
    Calculate $\alpha$ s.t. split width is equal to $w|x_k^{max} - x_k^{min}|$
    Create fuzzy split at root of $T$ with parameters $\alpha, A_k, c$ using Eq.4
    Fuzzify($T \to$ left)
    Fuzzify($T \to$ right)
Return $T$

Optimize_overlap_width_parameter($E, S$)
Input: $E$ - an ensemble, and $S$ - a training set
Output: optimal overlap width
Let $w = [10\%, 20\%, 30\%, .., 90\%]^T$
For $k = 1 \to 9$ do
    For each tree $T$ in ensemble $E$ do
        $T$ = Fuzzify($T, w_k$)
    Let $err_k$ be a cross-validation estimate of the error of ensemble $E$, using set $S$
Let $p = \underset{k}{argmin}(err_k)$
Return $w_p$
```

(NN). We use feed-forward multi-layer perceptron (MLP) networks [1] with one hidden layer of nodes. The only parameter that we tune for NN is the number of nodes in the single hidden layer. We test values for this parameter in the range from 1 to 10, so in total we test 10 possible values. For this work we don't anticipate a need for more than 10 hidden neurons or the need for more than one hidden layer. The implementation of the multi-layer perceptron networks that we use is the one from the Neural Network Toolbox in Matlab.

The second method is the Adaptive Network Based Fuzzy Inference System (ANFIS) [14]. It is a hybrid neural-network approach, which combines backpropagation gradient descent and least-squares regression. The model it builds is a set of fuzzy rules (Takagi-Sugeno fuzzy model). The premise parts of the rules are built using fuzzy sets, while the consequents are linear models of the feature attributes.

The method first determines the number of fuzzy rules and their initial positions (structure identification part). This is done by using a fuzzy c-means clustering algorithm. After the initial stage, the method applies an iterative improvement of the model, consisting of two steps. The first step is backpropagation gradient descent, which modifies the parameters of the fuzzy membership functions (it re-locates the centers of the fuzzy rules and changes the parameters of the Gaussian membership functions). The second step is learning the parameters of the linear models of the rule consequents. For this step a weighted least-squares regression is applied, and the linear model parameters of all rules are learnt simultaneously.

The only parameter that needs to be set for ANFIS is the number of rules (clusters). The Matlab implementation (available in the Fuzzy Logic Toolbox) we use has the option to determine automatically the number of rules, but in our experience suboptimal results are obtained in that way. We thus tune the value of this parameter (which is in the range from 2 to 10) by considering 9 possible integer values for it.

The parameters that need tuning for MTE are the following a) the minimal number of examples t, b) the attribute randomization parameter p and c) the overlap width parameter w. First, we perform a search for the optimal combination of the two parameters (t_{opt}, p_{opt}) using internal cross-validation of ensembles of 10 trees. Once this pair has been found, we optimize the overlap width parameter w. We consider 9 different values for this parameter, ranging from 10% to 90%. After the parameter optimization step, the training of the ensembles is carried out, by learning 50 trees. The total number of parameter combinations tried for each of the datasets is reported in Section 4.2.

We also consider four alternatives of our method: bagging of model trees (BMT), bagging of fuzzified model trees (BTM+fz), (random) forests of model trees (FMT) and (random) forests of fuzzified model trees (FMT). In addition, we consider several baselines closely related to our MTE method. The baseline methods chosen are: a single M5′ model tree (M5′), bagging of M5′ regression trees (BRT) and (random) forests of M5′ regression trees (FRT). The parameters of the baseline methods are optimized using the same internal cross-validation procedure as for the MTE method. The baseline methods we consider are implemented in WEKA [11].

3.3 Evaluation Criterion and Methodology

We evaluate the predictive performance and training time of the MTE method and compare these to the corresponding measures for NN and ANFIS. For the predictive performance, we calculate the squared prediction error, and report the root-relative mean-squared error (RRMSE), as defined by (5).

$$RRMSE = \frac{\sqrt{\sum (y_i - \hat{f}(x_i))^2}}{\sqrt{\sum (y_i - \bar{y})^2}} \tag{5}$$

Note that RRMSE normalizes the root mean squared error (RMSE) by the error of the single model that always predicts the average.

For the training time, we evaluate the time needed for parameter optimization and training (i.e. learning a regression model). The times are reported in seconds and all of the experiments are run on a machine with an Intel Pentium 4 CPU running at 3.2Ghz and 2GB of RAM memory.

The parameter optimization part of the learning is carried out by using 5-fold cross-validation. The prediction error of the cross-validation procedure is a criterion for selecting the best parameter or parameter combination. In order to get a more reliable estimate of the performance of the methods with the selected parameters, we run the experiments 20 times using different random seeds. The performance values reported in Section 4 are the mean and standard deviations of the 20 experiment runs. Note that ANFIS uses randomization for setting the initial parameters of the fuzzy rules, NN use randomization to set the initial parameters of the network and MTEs use randomization for creating the bootstrap replicates and for split attribute randomization.

4 Results and Discussion

Here we report the experimental results, i.e. the predictive performance of the methods, the time for training needed, and the effect of the size of the ensembles (number of trees) on the predictive performance.

4.1 Predictive Performance

The predictive performance of the MTE method, its variants and the baseline methods we compare it to on the dynamic system datasets is shown in Table 5 and Figure 1. First we note that ensembles of regression trees perform much worse than all model-tree based methods. They perform worse than even a single model tree. This is in line with the perceived idea of smoothness and extrapolation for modeling dynamic systems. Note that bagging of model trees performs worse than an individual model tree, both for crisp and fuzzified trees. The only exception is ensembles of the pH20 datasets, where bagging of fuzzified trees performs better than a single model tree. Random forests of model trees (both crisp and fuzzy) perform better than bagging.

In fact, random forests of fuzzified model trees perform best for all three datasets. For the GLS and pH datasets, the improvement in predictive performance over that of a single tree is small. On the noisy dataset pH20, the improvement in performance is clearly visible. The fuzzification of the trees in the ensemble helps in decreasing the error, both for random forests and for bagging of model trees.

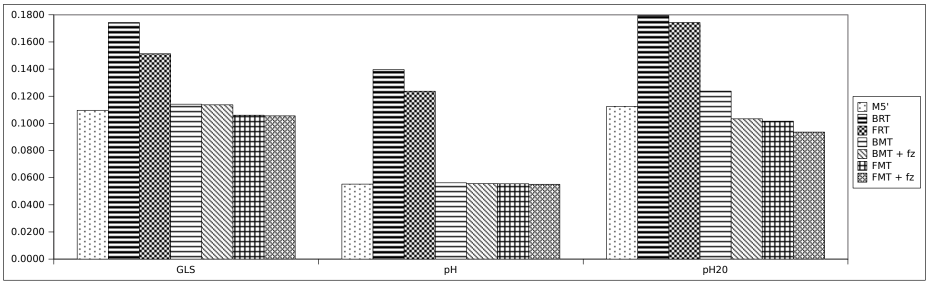


Fig. 1. Predictive performance of Model Tree Ensemble variants and baseline tree-based models

Table 5. Predictive performance of Model Tree Ensemble variants and baseline tree-based models

	GLS		pH		pH20	
	mean	st.dev.	mean	st.dev.	mean	st.dev.
M5′	0.1096	0.0000	0.0553	0.0000	0.1125	0.0000
BRT	0.1743	0.0017	0.1396	0.0030	0.1794	0.0048
FRT	0.1513	0.0036	0.1237	0.0091	0.1743	0.0049
BMT	0.1142	0.0056	0.0562	0.0021	0.1238	0.0027
BMT + fz	0.1137	0.0054	0.0556	0.0016	0.1033	0.0018
FMT	0.1060	0.0019	0.0555	0.0020	0.1016	0.0029
FMT + fz	**0.1055**	0.0019	**0.0551**	0.0017	**0.0936**	0.0018

The predictive performance of the MTE methodology is also compared to the selected methods, used in system identification, and described in Section 3.2. The results of the comparison are shown in Table 6 and Figure 2.

Table 6. Predictive performance of MTEs as compared to selected methods used in system identification

	GLS		pH		pH20	
	mean	st.dev.	mean	st.dev.	mean	st.dev.
FMT	0.1060	0.0019	0.0555	0.0020	0.1016	0.0029
FMT + fz	0.1055	0.0019	**0.0551**	0.0017	**0.0936**	0.0018
ANFIS	**0.1015**	0.0000	0.0751	0.0000	0.0937	0.0000
NN	0.1243	0.0350	0.0803	0.0095	0.1284	0.0139

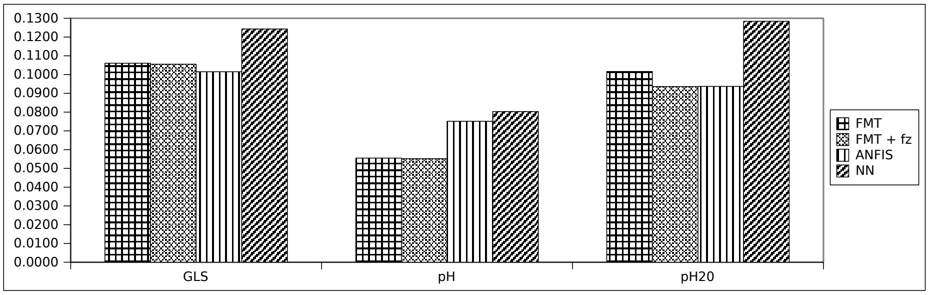


Fig. 2. Predictive performance of MTEs as compared to selected methods used in system identification

The comparison of the RRMSE for MTEs, ANFIS and NN shows that the performance of MTEs is better or comparable to ANFIS. On the GLS dataset, ANFIS is the winning method, but MTEs are the winning method in the other two cases. The performance of ANFIS is worse on the pH dataset, as the optimal number of fuzzy rules, as determined by the cross-validation procedure, is only 2. For comparison, the optimal number of fuzzy rules for the noisy counterpart, i.e. pH20 dataset, is 5, which leads to more competitive performance.

For the noisy pH20 dataset, MTEs show good performance, comparable to that of ANFIS. The standard deviation of the estimates of RRMSE error is almost zero for ANFIS and quite small for the MTE method. On the other hand, Neural Networks show quite a large deviation of the RRMSE result, because the random initial values used for the optimization of the network parameters lead to solutions with very different quality.

The optimal sets of parameters for the results presented in Tables 5 and 6 are as follows: For the FMT+fz variant of the MTEs, the value of the parameter t, the minimal number of examples, was 125, 50 and 90 for the three datasets, i.e. GLS, pH and pH20, respectively. The value of the attribute randomization parameter p was 1, 2 and 2, while the optimal overlap w had values of 20%, 20% and 30%. The number of trees included in all of the ensembles was 50. The optimal number of rules for ANFIS was 2, 2 and 5, while the optimal number of nodes in the hidden layer of NN was 6, 2 and 2 for the three datasets, respectively.

4.2 Computing Times

In this section, the time for parameter optimization and training is evaluated both separately and together for the three methods: MTE, ANFIS and NN. Table 7 summarizes the number of parameters that need tuning for each method and the number of possible combinations of parameter values that were tested in the 5-fold cross-validation procedure. It also reports the time needed for parameter optimization and for training (separately and together).

As discussed in Section 3.2 the MTE methodology has three parameters to tune and the total number of parameter value combinations tested ranges from 72 to 120. However, the tree building procedure is relatively fast, so the total time for model selection comparable to the time taken by the two selected methods. The results show that, in spite of the large number of parameter value combinations tested for MTEs, it takes less time to determine their optimal values and train the resulting model as compared to ANFIS. It can be noticed that the duration of the training procedure is shortest for NN, while it is longest for ANFIS. This is also visible in Figure 3.

Table 7. Computing times taken for parameter optimization and model training. The number of parameters tuned and the number of combinations of their values are listed as well.

		Parameter optimization			Training	Total
		time (sec)	num.param.	num.values	time (sec)	time (sec)
GLS	FMT+fz	230.0	3	120	2.6	232.6
	ANFIS	319.0	1	9	17.5	336.5
	NN	174.6	1	10	4.1	178.7
pH	FMT+fz	72.0	3	72	1.1	73.1
	ANFIS	197.5	1	9	7.5	205.0
	NN	45.9	1	10	1.0	46.9
pH20	FMT+fz	79.1	3	72	0.8	79.9
	ANFIS	198.1	1	9	7.7	205.8
	NN	33.8	1	10	0.6	34.4

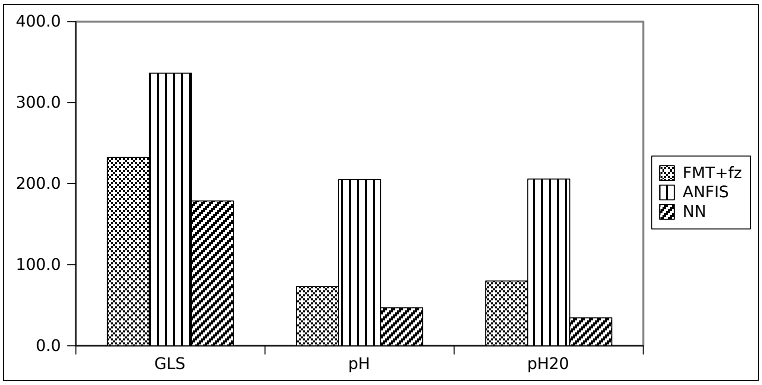


Fig. 3. Computing times

Note that decision trees scale very well as the number of training instances and the number of features increases. The same holds for ensembles of trees, especially for random forests. We thus expect our MTE methodology to perform more efficiently than both ANFIS and NN for larger problems.

4.3 The Effect of the Number of Trees in Ensemble

Table 8 and Figure 4 present the predictive performance (in terms of RRMSE) of MTEs with different number of trees. The results, obtained by using the FMT variant, show that the mean performance (out of 20 runs) increases slightly but not substantially by using more trees. This means that the optimal predictive performance can be obtained by using as few as 10 or 50 trees. In Table 8 however, it can be seen that the standard deviation of the error decreases when more trees are included in the ensemble.

Table 8. Effect of the number of trees in the ensemble on its performance

Num.trees in ensemble	GLS		pH		pH20	
	mean	st.dev.	mean	st.dev.	mean	st.dev.
10	0.1091	0.0044	0.0566	0.0032	0.1038	0.0059
50	0.1060	0.0019	0.0555	0.0020	0.1016	0.0029
100	0.1053	0.0020	0.0558	0.0011	0.1012	0.0025
500	0.1054	0.0007	0.0553	0.0006	0.1002	0.0010

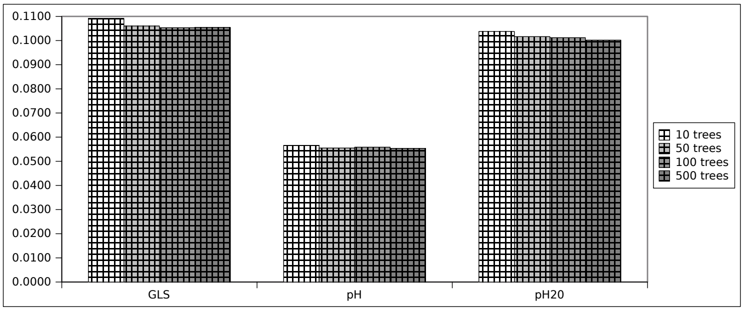


Fig. 4. Effect of the number of trees in the ensemble on its performance

5 Conclusions

In this paper, we investigated the task of modeling dynamic systems using tree ensembles. As it requires a modeling procedure which produces a smooth fit and a reasonable extrapolation behavior, we experimentally confirmed our hypothesis that regression trees and ensembles thereof are not appropriate. Instead, we introduced and evaluated ensembles of model trees with fuzzy splits.

Our experimental evaluation showed that the fuzzification procedure improves the performance of the model tree ensembles, especially on noisy data. Overall, the accuracy of the model tree ensembles was better than ensembles of regression trees and it also was better than a single model tree. Compared to selected state-of-the-art methods used in the area of system identification, the model tree ensembles performed equally well or better.

In our future work, we would like to evaluate the performance of ensemble models using a more stringent evaluation methodology tailored for modeling dynamic systems. This evaluation concerns the output error which is calculated by a procedure of simulation: the model at hand is repeatedly used for prediction in consecutive time points, with the feature vector at each time point modified to contain previous model predictions instead of the measured lagged values of the system variable. The evaluation in terms of output error would show whether the model predictions can be used for practical purposes or overfit noise and cause the simulation to diverge.

Acknowledgements. We would like to gratefully acknowledge the financial support of the following institutions: The Slovene Human Resources Development and Scholarship Fund, the Slovenian Research Agency (Grants P2-0001 and P2-0103), the European Commission (Grants ICT-2010-266722 and ICT-2011-287713), and the Operation no. OP13.1.1.2.02.0005 financed by the European Regional Development Fund (85%) and the Ministry of Education, Science, and Sport of Slovenia (15%).

References

1. Nelles, O.: Nonlinear system identification: from classical approaches to neural networks and fuzzy models. Springer (2001)
2. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press (2000)
3. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press (2006)
4. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
5. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
6. Murray-Smith, R., Johansen, T. (eds.): Multiple Model Approaches to Modelling and Control. Taylor and Francis systems and control book series. Taylor and Francis, London (1997)
7. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine Learning 63(1), 3–42 (2006)
8. Breiman, L.: Randomizing outputs to increase prediction accuracy. Machine Learning 40(3), 229–242 (2000)
9. Wang, Y., Witten, I.H.: Inducing model trees for continuous classes. Poster Papers of the 9th European Conference on Machine Learning (ECML 1997), Prague, Czech Republic, pp. 128–137 (1997)
10. Quinlan, J.R.: Learning with continuous classes. In: Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, vol. 92, pp. 343–348 (1992)
11. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. ACM SIGKDD Explorations 11(1), 10–18 (2009)
12. Kocijan, J., Likar, B.: Gas–liquid separator modelling and simulation with Gaussian-process models. Simulation Modelling Practice and Theory 16(8), 910–922 (2008)
13. Henson, M.A., Seborg, D.E.: Adaptive nonlinear control of a pH neutralization process. IEEE Transactions on Control Systems Technology 2(3), 169–182 (1994)
14. Jang, J.S.R., Sun, C.T., Mizutani, E.: Neuro-Fuzzy and Soft Computing–A Computational Approach to Learning and Machine Intelligence. Prentice Hall (1997)

Fast and Scalable Image Retrieval Using Predictive Clustering Trees

Ivica Dimitrovski[1], Dragi Kocev[2], Suzana Loskovska[1], and Sašo Džeroski[2]

[1] Faculty of Computer Science and Engineering, University of Ss Cyril and Methodius
Rugjer Boshkovikj 16, 1000 Skopje, Macedonia
[2] Department of Knowledge Technologies, Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
{ivica.dimitrovski,suzana.loshkovska}@finki.ukim.mk,
{Dragi.Kocev,Saso.Dzeroski}@ijs.si

Abstract. The recent overwhelming increase in the amount of available visual information, especially digital images,has brought up a pressing need to develop efficient and accurate systems for image retrieval. State-of-the-art systems for image retrieval use the bag-of-visual-words representation of the images. However, the computational bottleneck in all such systems is the construction of the visual vocabulary (i.e., how to obtain the visual words). This is typically performed by clustering hundreds of thousands or millions of local descriptors, where the resulting clusters correspond to visual words. Each image is then represented by a histogram of the distribution of its local descriptors throughout the vocabulary. The major issue in the retrieval systems is that by increasing the sizes of the image databases, the number of local descriptors to be clustered increases rapidly: Thus, using conventional clustering techniques is infeasible. Considering this, we propose to construct the visual codebook by using predictive clustering trees, which are very efficient and have good performance. Moreover, to increase the stability of the model, we propose to use random forests of predictive clustering trees. We evaluate the proposed method on a benchmark database of a million images and compare it to other state-of-the-art methods. The results reveal that the proposed method produces a visual vocabulary with superior discriminative power and thus better retrieval performance.

Keywords: image retrieval, visual vocabulary construction, predictive clustering.

1 Introduction

An ever increasing amount of visual information is becoming available in various digital archives. For instance, the widely used social web sites such as FACEBOOK[1] and FLICKR[2] store several billions images for their users. The improvement of digital cameras and user interfaces for upload of images will further

[1] Facebook©– http://www.facebook.com
[2] Flickr from Yahoo!©– http://www.flickr.com

increase the amount of available images. The value of the information obtained from an image depends on how easily it can be found, retrieved, accessed, filtered and managed. Considering this, the development of systems for efficient archiving, browsing and searching images is a necessity. Such systems are being developed within the research area of image retrieval.

Image retrieval is an inter-disciplinary research area that cross-fertilizes the following research areas: multimedia research, information retrieval, machine learning, computer vision, and human-computer interaction. The methods for image retrieval can be categorized into two categories [1]: text-based image retrieval (TBIR) and content-based image retrieval (CBIR). The former group of methods require some meta-data for each image in textual format (i.e., image tags) and then the retrieval is performed by providing textual queries. These methods perform well and are efficient as long as the images are correctly tagged. However, these methods have two serious limitations: a large amount of human labour for manual annotation (which is even more exacerbated in the case of large image databases) and the inaccuracy from the subjectivity of the human annotators. To alleviate these limitations, CBIR methods were introduced. They describe the images by their visual content, such as color, texture, shapes and local descriptors. The CBIR methods heavily rely on extracting appropriate image descriptors and a good similarity measure between images.

In this work, we focus on developing a method for efficient CBIR in large scale image databases. More specifically, we are concerned with developing a method for particular object retrieval from a large scale database which will retrieve all images that contain the specific query object. In other words, we are interested to associate two images based on the objects they contain and not based on the entire images. For example, if the query object is a specific model of a BMW car, then the method should retrieve the images containing that specific model of a BMW and not other models of a BMW or any other type of car.

These methods can be readily applied in several practically relevant domains [2]. To begin with, they can be used for creation of a web-scale visual search engine where the query will be a visual object. Second, they can be used for searching through personal image databases (e.g., select the photos that contain an image of the Eiffel Tower). Next, performing product search will strongly benefit from such systems: The user can take a photo of a given product, perform a search on the web and compare its prices from the given store to the price in the on-line shops. Furthermore, these methods will facilitate automatic tagging of images upload on social media, such as Flickr and Facebook. Finally, these methods can be used for augmented reality and creation of visual guides for museums and art galleries. For instance, a user can take a photo of a given sculpture and look for info on the web for the given sculpture.

Several systems for particular object retrieval have been proposed in the literature [3,2,4]. These systems are inspired from the text retrieval systems using the analogy between bag-of-words and bag-of-visual-words representation [5]. They consist of three phases: creation of visual vocabulary, image description and similarity definition. The creation of the visual vocabulary starts with detection of interesting points in the images. From these points, then, local invariant descriptors are extracted. Finally, the visual codebook is obtained by clustering the large set of descriptors obtained from all of the images. The resulting clusters represent the visual word, while all the visual words comprise the visual dictionary. The image description phase consists of assigning all of the local image descriptors to the visual words from the visual dictionary. Each image is then described with a high-dimensional histogram and each component from the histogram is the number of descriptors that are assigned to a given visual word. Finally, the images are ranked using term frequency inverse document frequency ($tf - idf$) scores which discount the influence of visual words which occur in many images. The search is then performed efficiently using a fast and tree-based inverted index structure [6].

The systems for particular object retrieval face several challenges [2]. To begin with, the changes in the lighting, image scale and rotation can hurt the performance of the retrieval systems. Second, viewpoint changes can make previously unseen parts of the object visible and also may include obstructions which will cover parts of the object. Finally, the systems need to be scalable with respect to the size of the image database, while preserving the retrieval accuracy. Moreover, the construction of the visual vocabulary should be also performed more efficiently.

In this paper, we present a novel method for fast and efficient construction of the visual codebook for particular object retrieval. The proposed method is based on the predictive clustering framework [7] which unifies predictive modelling and clustering through methods that partition the instances into subsets, such as decision trees and decision rules. The task of predictive clustering is to identify clusters of instances that are close to each other both in the target and in the descriptive space. In this work, we use the predictive clustering trees (PCTs) to construct the visual vocabulary. More specifically, we use PCTs for predicting multiple targets in which the descriptive attributes are also considered as clustering/target attributes.

Using a single PCT is a fast and efficient approach to the construction of visual codebooks. However, PCTs (and decision trees in general) are unstable, i.e., can change substantially for small changes in the data. To further improve the discriminative power and the robustness of our approach, we are using a small ensemble (random forest) of PCTs (as suggested in [8]). This produces several visual codebooks (each codebook corresponding to a single PCT from the ensemble). The overall visual codebook is then obtained by concatenating the visual codebooks from the single PCTs.

The remainder of this paper is organized as follows. Section 2 briefly presents the related state-of-the-art methods for particular object retrieval. The predictive clustering framework is described in Section 3. Section 4 gives the proposed method for codebook construction for particular object retrieval. Section 5 outlines the experimental design, while Section 6 presents the results from the experimental evaluation. Finally, the conclusions and a summary are given in Section 7.

2 Related Work

In this section, we briefly present the most widely used and state-of-the-art methods for CBIR. Many studies have shown that the bag of visual words approach exhibits a high retrieval performance for object retrieval given a query image of some particular object [9,10]. A crucial step in the bag of visual words approach is the codebook construction. The best results are achieved by using large codebook that contains one million or even more entries/visual words, which requires clustering tens or even hundreds of millions of high-dimensional feature descriptors into one million or more clusters. The methods for CBIR mainly differ in the process of the visual codebook construction.

The visual codebook is typically constructed by applying k-means clustering to the local descriptors (e.g., Scale Invariant Feature Transform descriptors) extracted from the images [11,12]. The resulting clusters are actually the visual words. Although this method works very well for smaller databases (and consequently small number of descriptors), it has very serious limitations when applied to the problem of large scale object retrieval [13]. It works with small visual codebooks, i.e., with only thousands of visual words, while many datasets may have tens of thousands of visual words.

The hierarchical k-means (HKM) approach [3] addresses the issue of small visual codebook. HKM performs the clustering in a hierarchical manner with a predefined number of levels (n). At the first level, the descriptor space is clustered into k_1 clusters, then at the second level, each of the k_1 clusters is re-clustered into k_2 clusters, and so on, until level n. The final visual codebook then consists of $k_1 \cdot k_2 \cdot ... \cdot k_n$ visual words. However, a major drawback of this method is that it is not a priori clear how many levels to use and how to choose the appropriate values for $k_1 ..., k_n$.

Philbin [2] proposed the approximate k-means (AKM) algorithm. In AKM, the exact nearest neighbor search is replaced with approximate nearest neighbor search in the assignment step when searching for the nearest cluster center for each point. In particular, the current cluster centers in each k-means iteration are organized by a forest of $k - d$ trees to perform an accelerated approximate nearest neighbor search.

Most closely related to our approach is the CBIR method based on indexing random sub-windows (extracted from the images) with extremely randomized trees [14]. These sub-windows are sampled from the images at random positions and random sizes. The sampled sub-windows are resized using bilinear interpolation to size 16×16. Each of the sub-windows is then described with the HSV values (thus resulting in a 768 feature vectors). Next, these feature vectors are used to construct extremely randomized tree: each node split is selected randomly; thus, these trees are constructed in an unsupervised manner. These trees are then used as search structures for the retrieval phase. Furthermore, the extremely randomized trees method can be used to construct ensembles thus further improve their retrieval performance [8].

Uijlings et al. [15] have performed experimental comparisons of visual dictionaries constructed using k-means and tree-based approaches. The main conclusions from their study is that the tree-based approaches are more efficient than approaches based on k-means. However, the improvement of the computational efficiency comes with the price of decreasing the discriminative power of the vocabulary.

In this paper, we propose a method for visual codebook construction using the predictive clustering framework. More specifically, we propose to construct the visual codebook using PCTs and random forests of PCTs. There are two major differences between the method proposed here and the one proposed by Mareé et al. [14]. First, the former is used in the vocabulary construction phase and then it employs the $tf - idf$ scores and inverted index for the retrieval process, while the latter method tries to emulate the complete process. Second, the selection of splits in the former is performed with an informed approach, while in the latter method this is done randomly. Namely, the split selection in the proposed method is performed by considering the compactness of the produced clusters. This split selection is the main reason for obtaining a visual vocabulary with high discriminative power.

3 Predictive Clustering

In this section, we first outline the predictive clustering framework, which is the foundation of our codebook generation approach. We then briefly describe the predictive clustering trees for predicting multiple targets. Finally, we present the method for construction of random forests of predictive clustering trees.

3.1 Predictive Clustering Framework

The notion of *predictive clustering* was first introduced by Blockeel [7]. The predictive clustering framework unifies two machine learning techniques, predictive modelling and clustering, usually viewed as completely different. The connection between these techniques is made by machine learning methods that partition the instances into subsets, such as decision trees and decision rules. These methods can be considered both as predictive and as clustering methods.

The task of predictive clustering is to identify clusters of instances that are close to each other both in the target and in the descriptive space. Figure 1 illustrates the tasks of predictive modelling (Figure 1(a)), clustering (Figure 1(b)) and predictive clustering (Figure 1(c)). Note that Figure 1 presents the target and the descriptive space as one-dimensional axes for easier visual interpretation, but they are usually of higher dimensionality.

The clusters that were obtained using the target space only (Figure 1(a)) are homogeneous in the target space (the target variables of the instances belonging to the same cluster have similar values). On the other hand, the clusters obtained using the descriptive space only (Figure 1(b)) are homogeneous in the descriptive space (the descriptive variables of the instances belonging to the same cluster have similar values). The predictive clustering combines these two and produces clusters that are homogeneous both in the target and in the descriptive space (Figure 1(c)).

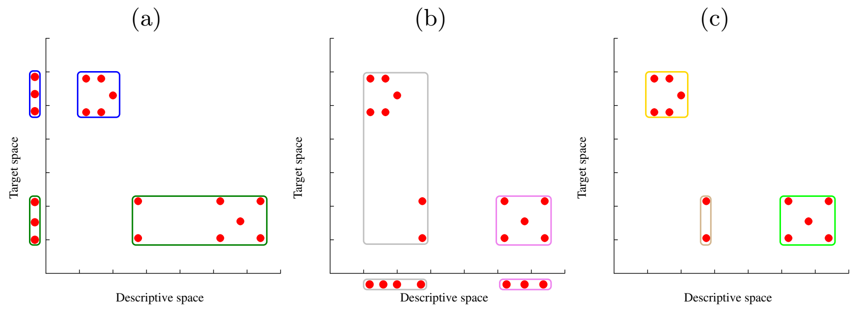


Fig. 1. An illustration of predictive clustering: (a) clustering in the target space, (b) clustering in the descriptive space, and (c) clustering in both the target and the descriptive space. Figure adapted from [7].

In this work, we use a specific setting from the predictive clustering framework where the descriptive space is equal to the target space, i.e., the target variables are used to provide descriptions for the obtained clusters. This focuses the predictive clustering setting more on the task of clustering. This approach has two major advantages over classical clustering (such as k-means). First, we obtain the clusters much more efficiently as compared to standard clustering algorithms. Second, there are cluster descriptions for each of the clusters. The cluster description is the conjunction of the tests starting from the root node of the tree then following the path to the leaf (or the conditions used in a predictive clustering rule). This also improves the efficiency when new examples need to be projected into the clusters.

3.2 PCTs for Multiple Continuous Variables

The predictive clustering framework is implemented using decision trees (called predictive clustering trees - PCTs) and decision rules (called predictive clustering rules) as predictive models. The predictive clustering framework sees a decision tree as a hierarchy of clusters: the top-node corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree. PCTs can be induced with a standard *top-down induction of decision trees* (TDIDT) algorithm [16]. The algorithm is presented in Table 1. It takes as input a set of examples (E) and outputs a tree. The heuristic (h) that is used for selecting the tests (t) is the reduction in variance caused by partitioning ($\mathcal{P}$) the instances (see line 4 of BestTest procedure in Table 1). By maximizing the variance reduction the cluster homogeneity is maximized and it improves the predictive performance. If no acceptable test can be found (see line 6), that is, if the test does not significantly reduces the variance, then the algorithm creates a leaf and computes the prototype of the instances belonging to that leaf.

The main difference between the algorithm for learning PCTs and a standard decision tree learner is that the former considers the variance function and the

Table 1. The top-down induction algorithm for PCTs

```
procedure PCT($E$) returns tree
1: $(t^*, h^*, \mathcal{P}^*) = \text{BestTest}(E)$
2: if $t^* \neq none$ then
3:     for each $E_i \in \mathcal{P}^*$ do
4:         $tree_i = \text{PCT}(E_i)$
5:     return node($t^*$, $\bigcup_i \{tree_i\}$)
6: else
7:     return leaf(Prototype($E$))
```

```
procedure BestTest($E$)
1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: for each possible test $t$ do
3:     $\mathcal{P}$ = partition induced by $t$ on $E$
4:     $h = Var(E) - \sum_{E_i \in \mathcal{P}} \frac{|E_i|}{|E|} Var(E_i)$
5:     if $(h > h^*) \wedge \text{Acceptable}(t, \mathcal{P})$ then
6:         $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: return $(t^*, h^*, \mathcal{P}^*)$
```

prototype function, that computes a label for each leaf, as parameters that can be instantiated for a given learning task. So far, the predictive clustering framework has been used for the prediction of multiple continuous variables, prediction of multiple discrete variables, hierarchical multi-label classification (HMC) and prediction of time series [17]. The predictive clustering framework is implemented in the CLUS system[3].

The variance and prototype functions of PCTs for predicting multiple continuous variables are instantiated as follows. The variance is calculated as the sum of the variances of the target variables, i.e., $Var(E) = \sum_{i=1}^{T} Var(Y_i)$. The variances of the target variables are normalized, so that each target variable contributes equally to the overall variance. This is due to the fact that the target variables can have completely different ranges. Namely, if one of the target variables is in the range (0, 1) and another in the range (10, 100) and normalization is not used, then the values of the second variable will contribute much more to the overall score than the values of the first variable. In addition, weighting of the (normalized values of the) target variables so that the variance function gives more weight to some variables and less to others is supported. The prototype function (calculated at each leaf) returns as a prediction the tuple with the mean values of the target variables, calculated by using the training instances that belong to the given leaf.

3.3 Random Forests of Predictive Clustering Trees

An ensemble classifier is a set of base classifiers, which typically has a better performance than the individual classifiers. A new example is classified by combining the predictions of each classifier from the ensemble for that example. The predictions are typically combined by taking their average (for regression tasks) or their majority/probability vote (for classification tasks).

We use the random forests method to create the base classifiers in the ensembles. A random forest [18] is an ensemble of trees, obtained both by bootstrap sampling of the training set and by randomly changing the feature set during learning. More precisely, at each node in the decision tree, a random subset of

[3] The CLUS system is available for download at `http://clus.sourceforge.net/`.

the input attributes is taken, and the best feature is selected from this subset (instead of the set of all attributes). The number of attributes that are retained is given by a function f of the total number of input attributes x (e.g., $f(x) = x, f(x) = \sqrt{x}, f(x) = \lfloor \log_2 x \rfloor + 1$).

4 Codebook Construction Using Predictive Clustering

The architecture of the system for fast and scalable image retrieval using PCTs is presented in Figure 2. The systems consists of an off-line phase and an on-line phase. The off-line phase implements the construction of the visual codebook, the image descriptions and the search structure for the retrieval. The on-line phase implements the construction of the query image description, the querying of the images and presenting the result of the retrieval. In the remainder of this section, we discuss these phases in more detail.

The off-line phase starts with the generation of the local descriptors for the images. For each image in the database, affine-invariant Hessian regions are located [2]. Typically there are 3300 regions detected on an image of size 1024×768. For each of these affine regions, a 128-dimensional Scale invariant feature transform (SIFT) descriptor is then computed [12]. Next, the descriptors are used to create the visual codebook, which is the central part of the image retrieval system.

The proposed method for constructing the visual codebook is as follows. First, we randomly select a subset of the local (SIFT) descriptors from all of the images. Next, the selected local descriptors constitute the training set used to construct a PCT. For the construction of the PCT, we set the descriptive attributes (i.e., the 128 dimensional vector) to be also target and clustering attributes. Note that, this feature is a unique characteristic of the predictive clustering framework. To control the size of the visual codebook, we apply pre-pruning of the trees by requiring a given minimum number of instances/descriptors in each tree leaf. In order to get the desired number of leaves of a tree (i.e., visual words) for a given dataset, the number of required instances in a leaf can be easily estimated (roughly, it should be a bit smaller than the ratio between the number of training examples and the desired codebook size). Each leaf of the tree is a separate visual word and all of the leaves constitute the visual codebook. After the construction of the visual codebook, we sort all of the descriptors through the tree and count the number of descriptors that fall in a given leaf (i.e., correspond to a given visual word). We describe each image then with a histogram (sparse frequency vector) of the number of descriptors per visual word.

The PCTs are computationally efficient: it is very fast to construct them and to produce a prediction. However, the trees are unstable, i.e., the structure of the tree can change substantially for small changes in the training data [18]. To overcome this limitation and to further improve the discriminative power of the visual codebook, we use a small random forest that consists of four PCTs (similarly as in [8]). The final visual codebook is then obtained by concatenating the individual visual codebooks from each of the PCTs in the forest, thus the size of the final visual codebook is the sum of the sizes of the individual visual codebooks.

Once the visual codebook is constructed and the image descriptors are obtained, we proceed to create the search structure for the retrieval. For the search engine, we use the vector-space model of information retrieval [6]. The query and each image in the database is represented as a sparse vector of visual word occurrences. The search then calculates the similarity between the query vector and each image vector by using a L_2 distance. As a weighting scheme for the distance, we use the standard $tf - idf$ weighting scheme [6], which reduces the contribution to the relevance score of the words that occur commonly (since they are less discriminative).

For computational efficiency, the search engine stores the word occurrences in an tree-like index structure. The index structure maps the individual words to the images in which they occur. In the worst case, the computational complexity of querying the index structure depends linearly from the database size, but in practice it depends closely to linear from the number of images that match a given query. This presents a big saving of computational time. For sparse queries, this can result in even a more substantial speed-up, as only images which contain visual words present in the query need to be examined. The scores for each image are accumulated so that they are identical to explicitly computing the similarity.

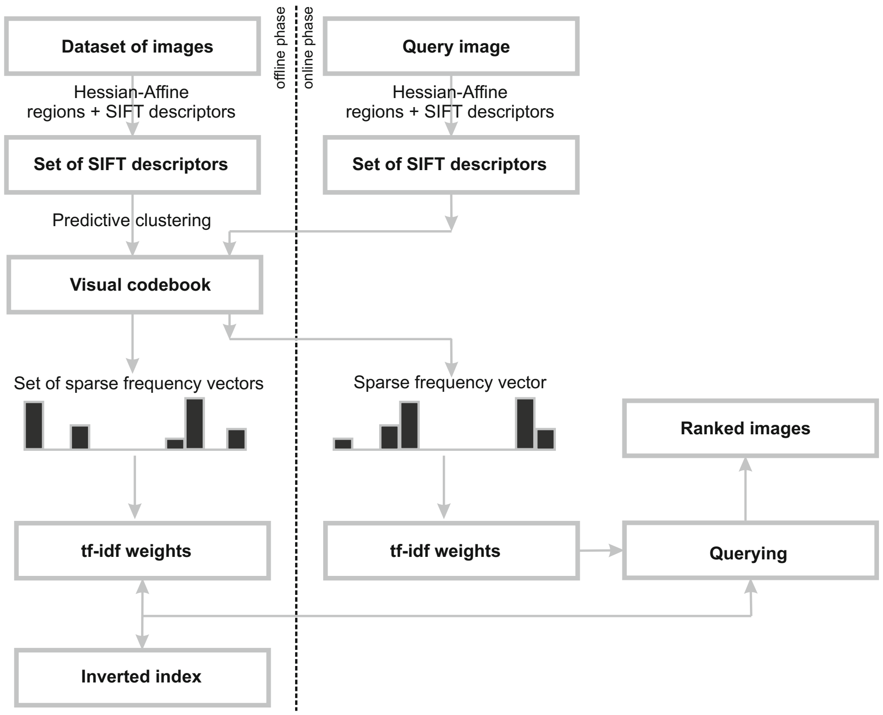


Fig. 2. Architecture of the proposed system for fast and scalable image retrieval based on predictive clustering trees

5 Experimental Setup

In this section, we present the experimental design we used to evaluate the proposed algorithm and compare it to other approaches. First, we present the dataset of images that we use. Next, we describe the evaluation metric we use to assess the retrieval performance. Finally, we state the experimental questions under investigation in this study.

5.1 Oxford Buildings Dataset

The Oxford Buildings dataset [19] is typically used as a benchmark dataset for large scale particular object retrieval. It consists of 5062 high-resolution images (1024×768) automatically downloaded from FLICKR by searching for 11 Oxford landmarks. The images are then manually annotated as to provide a solid ground truth for evaluation of the performance of retrieval systems. It also defines 55 queries (5 query objects/images for each of the 11 Oxford landmarks) that are used for performance evaluation. Each query consists of an image and query region/object of interest. This dataset is very challenging due to the substantial variations in scale, viewpoint and lighting conditions of the images and the objects that need to be retrieved.

In addition to this dataset often called Oxford $5K$, we use two other datasets to test the scaling abilities of the retrieval systems: Oxford $100K$ and Oxform $1M$ [2]. The Oxford $100K$ dataset contains 99782 high resolution images (1024×768), which were obtained from FLICKR by searching the 145 most popular tags. The Oxford $1M$ is created in a similar way, by crawling the 450 most popular tags from FLICKR and it consists of 1040801 images with medium resolution (500×333). These datasets are not challenging in terms of size, but also in the fact that they have much broader domain than the Oxford buildings: they include a mixture of different scenes, object, people, and buildings from all around the world.

5.2 Evaluation Measure

The most widely used performance measure for evaluation of methods in image retrieval is the *mean average precision* (mAP) [9,10]. We thus adopt mAP to evaluate the performance of our method for particular object retrieval and the discriminative power of its visual codebook.

The mAP is calculated as follows. For each of the 55 queries (5 for each of the 11 chosen Oxford landmarks), the average precision (AP) is computed as the area under the precision-recall curve ($AUPRC$) for that query. This score combines both precision and recall into a single performance score. Precision is defined as the ratio of retrieved positive images to the total number retrieved. Recall is defined as the ratio of the number of retrieved positive images to the total number of positive images in the dataset. An ideal precision-recall curve has precision 1 over all recall levels and this corresponds to an average precision of 1 and also $AUPRC$ of 1. The values of $AUPRC$ score are in the range [0,1] and if they are bigger then the retrieval performance of the system is better. The overall mAP value is obtained by taking the mean of the average precisions of all queries and it is used as a single number to evaluate the overall performance.

5.3 Experimental Questions

We focus the experimental evaluation of the propose method on the following three research questions:

1. Does the use of predictive clustering trees for visual codebook construction improves the retrieval performance of the bag-of-visual-words approach compared to the widely used methods based on k-means and approximate k-means?
2. Whether the increase of number of descriptors and the visual codebook size influences the retrieval performance?
3. Does the proposed method for visual codebook construction is efficient and scalable to larger problems?

In order to answer these three question, we designed the following experimental setup. For answering the first question, we compare the performance of the visual codebook constructed using the random forest of PCTs with the performance of the one constructed using k-means and approximate k-means. This comparison is justified by the fact that most current results related to content-based particular object retrieval are obtained by using large visual codebooks created using k-means, while the current state-of-the-art results are obtained using approximate k-means.

We address the second question by constructing visual codebooks with different size and by using different numbers of local descriptors. More specifically, we use $800K, 1M, 5M, 16.7M$ of local descriptors and produce codebooks with $10K$, $20K$, $50K$ and $1M$ visual words, respectively. The codebooks are constructed using our approach and the competing k-means and approximate k-means.

For answering the third question, we apply the visual codebook obtained by our method to the Oxford $100K$ and Oxford $1M$ image datasets. The performance results are compared with the results obtained using approximate k-means. We do not compared with exact k-means because it can't be scaled up to such a huge number of descriptors and visual words [2].

6 Results and Discussion

In this section, we present and discuss the results obtained from the experimental evaluation of the proposed method. First, we compare the performance of the proposed method to other methods from the literature. Next, we discuss the influence of the size of the codebook and the number of local descriptors considered to the retrieval performance of our system. Finally, we show the scalability of the proposed method by comparing its performance on a database with a million images.

Table 2 shows the results of the comparison of the three algorithms for visual codebook construction: k-means, approximate k-means (AKM) and random forest of predictive clustering trees (RF of PCTs). The results include experiments with a varying number of descriptors and a varying number of clusters. From the presented results, we can note that our method has a performance that is as least good as the one of the competing methods.

Table 2. Retrieval performance of k-means, approximate k-means and our algorithm based on predictive clustering trees (RF of PCTs) over the Oxford Buildings image dataset using different number of descriptors and visual words

Clustering parameters		mAP		
# of descriptors	codebook size	k-means	AKM	RF of PCTs
$800K$	$10K$	0.355	0.358	0.360
$1M$	$20K$	0.384	0.385	0.384
$5M$	$50K$	0.464	0.453	0.468
$16.7M$	$1M$	/	0.618	0.618

Furthermore, these results clearly show that the increase of the number of local descriptors and codebook size improve the retrieval performance of the systems. The best result (last row of Table 2) is obtained by using visual codebook with 1 million visual words obtained by all SIFT descriptors generated from all the images of the Oxford $5K$ dataset. To construct the codebook our method was running for $43.35h$, while the approximate k-means needed $69.9h$ (which is $\sim$ 1.6 times slower). Considering that the proposed method based on random forests of PCTs is very computationally efficient [17], we can at small computational expense construct even larger visual codebook. An initial set of experiments indicates that further increase of the codebook size improves even more the retrieval performance.

Next, we evaluate the scalability of our method on the $5K$, $5K + 100K$ and $5K + 100K + 1M$ datasets using the $1M$ words visual codebook. The results are given in Table 3. Here, we compare our method to approximate k-means, since using the standard k-means is not feasible. From the results, we can see that the retrieval performance of our method is better than the one of approximate k-means. We can also note the drop in performance with the increase of the size of the image database. This is mainly due to the fact that these datasets now include much more noisy images outside of the domain of Oxford buildings.

We visually inspect the retrieval results and illustrate part of them in Figures 3 and 4. The first image in each row is the image that contains the query object, while the remaining four images are part of the retrieved images. The retrieval results given in Figure 3 reveals that the proposed method performs very well when there are considerable variations in the viewpoint, the image scale, the lighting and partial occlusion of the object.

On the other hand, the retrieval results given in Figure 4 illustrate examples in which the proposed systems fails to successfully retrieve the particular query

Table 3. Comparison of the discriminative power (with the mAP values) of the visual codebooks obtained by AKM and our method (RF of PCTs) over the three image datasets (both the algorithms use a codebook with $1M$ visual words)

Image dataset	AKM	RF of PCTs
$5K$	0.618	0.618
$5K$+$100K$	0.490	0.512
$5K$+$100K$+$1M$	0.393	0.401

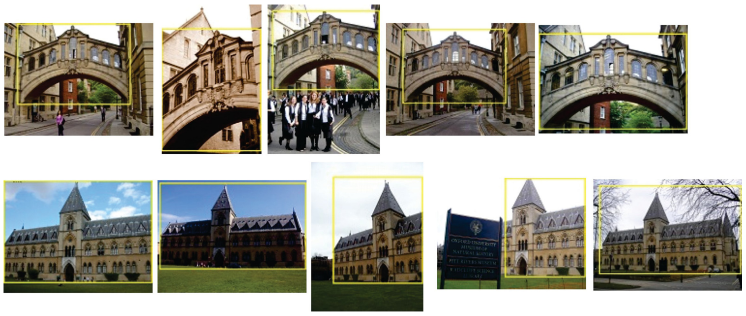


Fig. 3. Examples of searching the $5K$ dataset for: Bridge of sighs, Hertford College (first row), Pitt Rivers Museum (second row). First image in each row is the query image and the selected region with yellow color is the query object/building. The other four images in each row are result images obtained using the proposed system and algorithm.

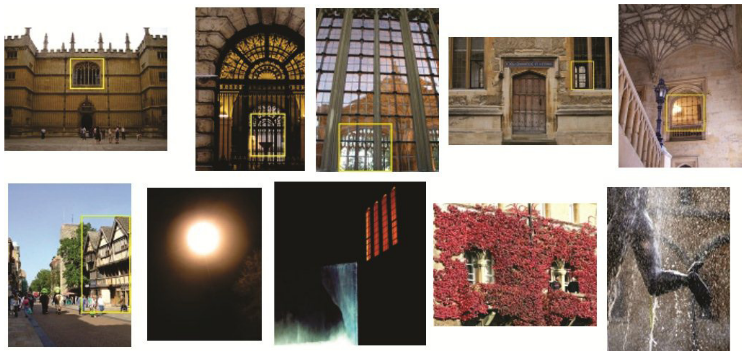


Fig. 4. Examples of errors in retrieval for two query images (first images in the rows). The false positives for the first query is visually plausible, but the false positive for the second query is due to visual ambiguities in the local regions and the low numbers of visual words in the retrieved image.

object. For the first query (the first row of images), the failure is due to the presence of images that are visually plausible and consistent with the query image (the retrieved images also contain windows with bars).

The failure for the second query (the second row of images) is a result of the visual ambiguities present in the images (the first two result images) and the 'burstiness' effect (the second two result images) [20]. The visual ambiguities arise because the images have a very small number of local regions. This in turn results in a very large $tf - idf$ weights in the matching phase thus making an error while retrieving the particular object. The burstiness effect appears when a given local descriptors appear more frequently in an image than a model would predict. In this context, the burstiness distorts the image similarity measure (i.e., the $tf - idf$ weights) and thus pollutes the ranking of the images in the retrieval results. In our case, this is a result of the rich texture present in the resulting images such as water-drops and flowers.

7 Conclusion

In this paper, we present a method for fast and efficient construction of visual codebooks for particular object retrieval from large scale image databases. The construction of a codebook is an essential part of the bag-of-visual-words approach to image retrieval. It should thus be efficient and deliver a codebook with high discriminative power. However, the construction of a visual codebook is a bottleneck in the bag-of-visual-words approach, because it typically uses k-means clustering over millions image patches to obtain several tens of thousands visual words. Existing approaches are able to solve the efficiency issue, however, a part of the discriminative power of the codebook is sacrificed for better efficiency. In this paper, we propose to use predictive clustering trees (PCTs) for codebook construction. In this way, we efficiently construct visual codebooks and increase the discriminative power of the dictionary.

PCTs are a generalization of decision trees and are capable of performing predictive modelling and clustering simultaneously. More specifically, the method we propose uses PCTs for predicting multiple targets to construct the visual codebook – each leaf in the tree is a visual word. Furthermore, we construct a small random forest of PCTs to increase the stability of the codebook and its discriminative power. The overall codebook is obtained by concatenating the smaller codebooks from each tree.

We evaluated the proposed method on the Oxford buildings image database, which is a benchmark database for large scale particular object retrieval. We used three variants of the database that include $5K$, $100K$ and $1M$ images. We compare the proposed method to literature standard and state-of-the-art methods: k-means and approximate k-means clustering.

The results from the experimental evaluation reveal the following. To begin with, our method has a performance that is as least good as the competing methods on the smaller database with $5K$ images. Next, the increase of the number of local descriptors and codebook size improve the retrieval performance of the systems. Considering that the proposed method is computationally efficient, we can afford to construct larger codebooks that in turn will increase the retrieval performance. Finally, on the large databases, our method exhibits better retrieval performance than the competing approximate k-means. All in all, the proposed method is able to efficiently produce a visual codebook with a high discriminative power.

We plan to extend this work along three major dimensions. First, we plan to extend the method and allow soft assignment of the descriptors to the visual words. Several studies have suggested that this could increase the retrieval performance of the system. Second, we will use the PCTs as search structures for performing the retrieval itself. This means that we will bypass the $tf - idf$ calculation and the creation of the inverted index, thus speed-up the retrieval even more. Finally, we will address the issue of burstiness by performing re-ranking of the top-ranked results by using spatial constraints. This procedure uses the predictions of the feature locations to estimate a transformation between the query region and each target image.

References

1. Liu, Y., Zhang, D., Lu, G., Ma, W.Y.: A survey of content-based image retrieval with high-level semantics. Pattern Recognition 40(1), 262–282 (2007)
2. Philbin, J.: Scalable Object Retrieval in Very Large Image Collections. PhD thesis, University of Oxford, Oxford, UK (2010)
3. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2161–2168 (2006)
4. Jégou, H., Harzallah, H., Schmid, C.: A contextual dissimilarity measure for accurate and efficient image search. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2007)
5. Sivic, J., Zisserman, A.: Video google: a text retrieval approach to object matching in videos. In: IEEE Conference on Computer Vision, pp. 1470–1477 (2003)
6. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press (1999)
7. Blockeel, H.: Top-down induction of first order logical decision trees. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium (1998)
8. Moosmann, F., Nowak, E., Jurie, F.: Randomized clustering forests for image classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(9), 1632–1646 (2008)
9. Everingham, M., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2010, VOC 2010 (2010), http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html
10. Nowak, S.: ImageCLEF@ICPR contest: Challenges, methodologies and results of the photo annotation task. In: International Conference on Pattern Recognition, pp. 489–492 (2010)
11. van de Sande, K., Gevers, T., Snoek, C.: Evaluating color fescriptors for object and scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(9), 1582–1596 (2010)
12. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
13. Jurie, F., Triggs, B.: Creating efficient codebooks for visual recognition. In: International Conference on Computer Vision, pp. 604–610 (2005)

14. Marée, R., Geurts, P., Wehenkel, L.: Content-based image retrieval by indexing random subwindows with randomized trees. In: Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H. (eds.) ACCV 2007, Part II. LNCS, vol. 4844, pp. 611–620. Springer, Heidelberg (2007)
15. Uijlings, J., Smeulders, A., Scha, R.: Real-time bag of words, approximately. In: ACM International Conference on Image and Video Retrieval, pp. 1–8 (2009)
16. Breiman, L., Friedman, J., Olshen, R., Stone, C.J.: Classification and Regression Trees. Chapman & Hall/CRC (1984)
17. Kocev, D., Vens, C., Struyf, J., Džeroski, S.: Tree ensembles for predicting structured outputs. Pattern Recognition 46(3), 817–833 (2013)
18. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
19. The Oxford Buildings Dataset (2013), `http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/`
20. Jégou, H., Douze, M., Schmid, C.: On the burstiness of visual elements. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1169–1176 (2009)

Avoiding Anomalies in Data Stream Learning

João Gama[1,2], Petr Kosina[1], and Ezilda Almeida[1]

[1] LIAAD-INESC TEC, University of Porto
{ezildacv,petr.kosina}@gmail.com
[2] Faculty of Economics, University Porto
jgama@fep.up.pt

Abstract. The presence of anomalies in data compromises data quality and can reduce the effectiveness of learning algorithms. Standard data mining methodologies refer to data cleaning as a pre-processing before the learning task. The problem of data cleaning is exacerbated when learning in the computational model of data streams. In this paper we present a streaming algorithm for learning classification rules able to detect contextual anomalies in the data. Contextual anomalies are surprising attribute values in the context defined by the conditional part of the rule. For each example we compute the degree of anomaliness based on the probability of the attribute-values given the conditional part of the rule covering the example. The examples with high degree of anomaliness are signaled to the user and not used to train the classifier. The experimental evaluation in real-world data sets shows the ability to discover anomalous examples in the data. The main advantage of the proposed method is the ability to inform the context and explain why the anomaly occurs.

Keywords: Data Streams, Rule Learning, Anomaly Detection.

1 Motivation

The amount of digital data currently handled is huge. Our ability to collect huge amounts of detailed information is increasing exponentially. Nevertheless, data anomalies such as inconsistencies, missing values, outliers, etc. are more frequent than desired. The existence of these data problems, commonly called *dirty data*, degrades the quality of the information with direct impact on the efficiency of data analysis techniques [1]. These problems can lead to incorrect decisions or strategies that often are costly to organizations. Improving data quality by detecting and eliminating errors and inconsistencies in the data is a relevant data mining problem.

Standard data mining methodologies define *pre-processing* as a key step before the learning phase. Pre-processing is essential to analyze the multivariate data sets before data mining. In this step, the data set is cleaned, by removing observations containing noise and those with missing data. The identification of observations that are not coherent with the rest of the data, can be used in two different perspectives. One perspective consists of removing these observations from the analysis. The other perspective considers these observations as interesting and therefore is very important to detect. Depending on the application, the user can be especially interested in the anomalous cases more than in the 'normal' observations. They may represent malicious cases such as intrusions, frauds or diseases. One way or another, the detection of anomalies is very important.

Most of the works in pre-processing data, anomaly and outlier detection are off-line. Current tasks usually consist of many observations being processed and the number is still increasing. It is increasing to the extent that the relatively recent approach called stream mining considers the data possibly infinite. Finding anomalies in such a setting is especially a difficult task not only because of the potentially unbounded size, but also because one of the typical characteristics of data stream is that the distribution generating the data can change over time.

This work presents a method for on-line anomaly detection, a crucial task in real-world applications. The method is embedded in a streaming classification rule learner, although it can be integrated with any VFDT like algorithm. The anomalies detected are characterized by a *context* that refers the region of the instance where the anomaly was detected, and behavioral attributes, those with anomalous values. This is a key advantage of the proposed system: it explains *where* and *why* a given example is anomalous.

The paper is organized as follows. The next section presents the related work in outlier and anomaly detection. Section 3 describes the method used to detect contextual anomalies that has been implemented inside a classification rule learner for data streams. Section 4 describes the anomalies detected in several well-known datasets. The last Section presents the conclusions and futures work.

2 Related Work

Anomaly detection refers to detecting observations that do not conform to an established normal behavior. Anomalies are also referred to as outliers, change, deviation, surprise, aberrant, peculiarity, intrusion, etc [2]. [9] points out the importance of data cleaning in developing real world applications. Data cleaning deals with missing values, noisy data, inconsistent data, etc. In this work, we focus in a particular form of inconsistent data: anomalies or outliers.

Statistical approaches were the earliest algorithms used for outlier detection. The most common approaches are univariate. Probably one of the simplest statistical outlier detection techniques use informal box plots [18] to pushup outliers in both univariate and multivariate data sets. Another single dimensional method was presented in [18] which calculates a Z value as the difference between the mean value for the attribute and the query value divided by the standard deviation for the attribute. The Z value for the query is compared with a 1% or 5% significance level. The technique requires no user parameters as all parameters are derived directly from data.

The literature in anomaly and outlier detection is huge. Two recent overviews, with excellent references are [11] and [2]. Most of the works refer to off-line approaches. A recent paper [15], addresses the anomaly detection problem in large-scale data mining applications using residual subspace analysis. The authors suggest a framework wherein random projection can be used to obtain compressed data. Their contribution shows that the spectral property of the compressed data is approximately preserved under such projection and thus the performance of spectral-based methods for anomaly detection is almost equivalent to the case in which the raw data is completely available.

[19] present an approach for combining adaptive pre-processing with adaptive online predictor. The authors present a case study with real sensory data from a production process. In that case, decoupling the adaptively of pre-processing and the predictor contributes to improving the prediction accuracy.

The authors of [2] define 2 types of anomalies.

- Point Anomalies. If an individual data instance can be considered as anomalous with respect to the rest of data, then the instance is termed as a point anomaly. This is the simplest type of anomaly and is the focus of majority of research on anomaly detection.
- Contextual Anomalies. If a data instance is anomalous in a specific context.. In this case, it is convenient to define:
  - Contextual attributes. The contextual attributes are used to determine the context for that instance.
  - Behavioral attributes. The attributes with abnormal values in the contexts defined by the contextual attributes.

A relevant aspect, pointed out by [2], is that an observation might be an anomaly in a given context, but an identical data instance (in terms of behavioral attributes) could be considered normal in a different context. This property is a key characteristic in identifying contextual and behavioral attributes for a contextual anomaly detection technique.

In [13], the authors discuss distance-based outlier detection methods for very large data bases, and propose several algorithms. The most efficient has complexity that is linear with the number of examples but exponential in the number of attributes. It is based on nearest neighbour search over cells defined by indexing structures. While the proposed algorithms are effective for very large data bases, its complexity limit their applicability in the streaming computational model.

One of the few systems that can detect contextual anomalies is Gritbot [16]. GritBot is not described in any scientific paper, is a commercial tool that detects inconsistencies in the data set. GritBot is an off-line tool that finds anomalies in data as a pre-processing to data mining algorithms. It can be thought as an autonomous data quality auditor that hunts for records having ”surprising” values of nominal and/or numeric attributes. Anomalies need not stand out in the complete dataset – GritBot searches for subsets of records in which the anomaly is apparent. Although there is no technical description of the methods used by GritBot, we can guess from the code, results and studies published by Quinlan that the GriBot generates rules iteratively. In each iteration, considers an attribute from a subset of n attributes as objective (dependent) attribute. Then, to each tuple that violates a certain rule is assigned the probability that the value anomaly can occur by chance and not by error. The approach we present in this paper identifies anomalies *a la* Gritbot, but on-line, with a single-scan over the data.

3 Anomaly Detection

The method we propose detects contextual anomalies. Contextual anomalies are characterized by a *context* that refers the region of the instance space where the anomaly was detected, and behavioral attributes, those with anomalous values. One example of the type of anomalies we detect, from the Adult dataset [5], is:

Case 15904:
```
education = 10th [6 in 1470]
capital-gain = 99999 [889.2±633.6]
```
Rule:
```
education-num <= 10 ∧
marital-status = Married-civ-spouse → >50K
```

The 15904th example is signaled as an anomaly, and is interpreted as follows. The context of the anomaly is given by the rule:
```
education-num <= 10 ∧
marital-status = Married-civ-spouse → >50K.
```
The attributes with suspicious values are *education* and *capital-gain*. The first attribute is nominal. In 1470 examples, the attribute value *education = 10* was observed 6 times. The second attribute is numerical. The mean of this variable (using the examples seen so far) is 889 and the standard deviation is 633. The anomaliness score for this example is 0.99.

In the first part of this section, we describe the algorithm to learn the decision rules defining the context of the anomalies. We should point out, that our anomaly detection system can be used in classification and regression problems with any VFDT like algorithms [4]. The current implementation is based on a stream classification rule learner, previously presented in [8]. In the next Section we provide a concise description of the learning algorithm, to clarify how the detection method works.

3.1 Very Fast Decision Rules Algorithm

As in many other systems, a rule in VFDR [14] is an implication of the form $A \Rightarrow C$. The A part of a rule is a conjunction of literals, that is, conditions based on attribute values. For numerical attributes, each literal is of the form $X_i > v$, or $X_i \leq v$ for some feature X_i and some constant v. For categorical attributes VFDR produce literals of the form $X_i = v_j$ where v_j is a value in the domain of X_i. The C part of a rule r, designated $\mathcal{L}_r$, is not a constant as in most of rule based systems, but a function. This is the most different feature of VFDR.

The VFDR algorithm is designed for high-speed data streams. It learns ordered or unordered rule sets. It needs only one scan of data and is able to provide any-time classifications.

Growing a Set of Rules. The algorithm begins with a empty rule set (RS) and a *default rule* $\{\} \rightarrow \mathcal{L}$, where $\mathcal{L}$ is initialized to $\emptyset$. $\mathcal{L}$ is a data structure that contains information used to classify test instances, and the sufficient statistics needed to expand the rule.

As already said, each learned rule (r) is a conjunction of literals, that are conditions based on attribute values, and a $\mathcal{L}_r$. If all the literals are true for a given example, then the example is said to be *covered* by the rule. The labeled examples covered by a rule

r are used to update $\mathcal{L}_r$. A rule is expanded with the literal that has the highest gain measure of the examples covered by the rule. $\mathcal{L}_r$ accumulates the sufficient statistics to compute the gain measure of all possible literals. $\mathcal{L}_r$ is a data structure that contains: an integer that stores the number of examples covered by the rule; a vector to compute $p(c_k)$, i.e., the probability of observing examples of class c_k; a matrix $p(X_i = v_j|c_k)$ to compute the probability of observing value v_j of a nominal attribute X_i per class; and a `btree` to compute the probability of observing values greater than v_j of continuous attribute X_i, $p(X_i > v_j|c_k)$, per class. The information maintained in $\mathcal{L}_r$ is similar to the sufficient statistics [6].

The number of observations, after which a rule can be expanded or new rule can be induced, is determined by the Hoeffding bound. It guarantees that, with probability at least $1 - \delta$, the true mean of a random variable x with a range R will not differ from the sample mean of size N by more than:

$$\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2N}}.$$

It is not efficient to check for the sufficient number of examples with every incoming example, therefore this is done only after every N_{min} observations.

The set of rules (RS) is learned in parallel as described in Algorithm 1. We consider two cases: learning ordered or unordered set of rules. In the former, every labeled example updates statistics of the first rule that covers it. In the latter, every labeled example updates statistics of all the rules that cover it. If a labeled example is not covered by any rule, the $default\ rule$ is updated.

The expansion of a rule is done using Algorithm 2 that employs the aforementioned Hoeffding bound. For each attribute X_i the value of split evaluation function G is computed for each attribute value v_j. If the best merit is better the second best with given confidence, i.e. satisfies condition $g_{best} - g_{2best} > \epsilon$, the rule is expanded with condition $X_a = v_j$ and the class of the rule is assigned according to the majority class of observations of $X_a = v_j$.

Classification Strategies. The set of rules learned by `VFDR` can employ different classification strategies: *First Hit*, and *Weighted Sum*. As in [3], the ordered rules use the *First Hit* strategy, while the unordered rules use the *Weighted Sum* strategy. In that case all rules covering the example are used for classification and the final class is decided by using weighted vote.

More specifically, assume that a rule r covers a test example. The example will be classified using the information in $\mathcal{L}_r$ of that rule. The simplest strategy uses the distribution of the classes stored in $\mathcal{L}_r$, and classify the example in the class that maximizes $p(c_k)$. This strategy only use the information about class distributions and does not look for the attribute-values, therefore it uses only a small part of the available information. In a more informed strategy, a test example is classified with the class that maximizes the posteriori probability given by Bayes rule assuming the independence of the attributes given the class. There is a simple motivation for this option. $\mathcal{L}$ stores information about the distribution of the attributes given the class usually for hundreds or even thousands of examples, before expanding the rule and re-initializing the counters. Naive Bayes (*NB*) takes into account not only the prior distribution of the classes,

Algorithm 1: VFDR: Rule Learning Algorithm

```
input : $S$: Stream of examples
        $N_{min}$: Minimum number of examples
        $ordered\_set$: boolean flag
output: $RS$: Set of Decision Rules
begin
    Let $RS \leftarrow \{\}$
    Let $default\ rule\ \mathcal{L} \leftarrow \emptyset$
    foreach example $(x, y_k) \in S$ do
        foreach Rule $r \in RS$ do
            if $r$ covers the example then
                Update sufficient statistics of Rule $r$
                if Number of examples in $\mathcal{L}_r$ mod $N_{min} = 0$ then
                    $r \leftarrow ExpandRule(r)$
                if ordered_set then
                    BREAK
        if none of the rules in RS trigger then
            Update sufficient statistics of the empty rule
            if Number of examples in $\mathcal{L}$ mod $N_{min} = 0$ then
                $RS \leftarrow RS \cup$ ExpandRule($default\ rule$)
```

but also the conditional probabilities of the attribute-values given the class. This way, there is a much better exploitation of the available information in each rule. Given the example $\boldsymbol{x} = (x_1, \ldots, x_j)$ and applying Bayes theorem, we obtain:

$$P(c_k|\boldsymbol{x}) \propto P(c_k) \prod P(x_j|c_k).$$

Using *NB* in VFDT like algorithms [4], is a well-known technique since it was introduced in [7]. One of its greatest advantages is the boost in any-time learning property because even though the learned rule set might not be robust enough or the individual rules might not provide sufficient information for expert interpretation (not being specialized enough, i.e., having only one or few conditions), it may already be able highly informed predictions based on *NB* classification.

3.2 Detecting Anomalies

Different kinds of rule systems are commonly used in multivariate anomaly detection. The use of AVFDR in on-line detection is one of the advantages the system provides. It can detect possible anomalies during the learning process. The detection process works as follows. When the system reads a new example, the rule set is checked to find the rules that cover the example. An example is covered by a rule, when the conditional tests of the antecedent of the rule are true for that example. For each attribute value, we compute the probability $P(X_i = v|Rule_r)$. These probabilities are computed from the

Algorithm 2: `ExpandRule`: Expanding one Rule

```
input : $r$: One Rule
        $G$: Split evaluation function;
        $\delta$: is one minus the desired probability
        of choosing the correct attribute;
output: $r$: Expanded Rule
begin
    Compute $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2N}}$ (Hoeffding bound)
    $EvaluateLiterals()$
    if $(g_{best} - g_{2best} > \epsilon)$ then
        Extend $r$ with a new condition based on the best attribute $X_a = v_j$
        Release sufficient statistics of $\mathcal{L}_r$
        $r \leftarrow r \cup \{X_a = v_j\}$
    return $r$
```

consequent of the rule, $\mathcal{L}_r$, that maintains the sufficient statistics required to expand the rule. Low values of these probabilities suggest that the example is an uncommon case in the context of the rule, and it is reported as anomaly. More specifically, for an example $(\boldsymbol{x}, y)$ and its attribute $X_i = v$ let

$$\Pr(X_i = v|\mathcal{L}_r)$$

be the probability of observing attribute value v of the attribute X_i given the conditions of a rule r.

We compute the univariate anomaliness score as:

$$Uscore_i = 1 - \Pr(X_i = v|\mathcal{L}_r) \tag{1}$$

This score is unsupervised in the sense that does not take into account the class label of the example. During the on-line learning, the learner receives labeled examples and therefore we can use the class information to compute the univariate score. The supervised univariate anomaly score is given by:

$$U^s score_i = 1 - \Pr(X_i = v|y, \mathcal{L}_r) \tag{2}$$

If $Uscore_i > \lambda$, for a given value of λ (typically 99%), the attribute value is said to be an anomaly for the context provided by rule r. This is applicable both for supervised and unsupervised scores.

Computing the Anomaly Score for Nominal Attributes. The domain of nominal is finite and unordered. For each nominal attribute, the statistics store in $\mathcal{L}_r$ of a rule, are in the form of a contingency table. For a given attribute, let N be the number of examples, seen so far, covered by rule r, $N_{i,\cdot}$ the number of examples, covered by rule r, where the attribute take the i^{th} value, $N_{\cdot,j}$ the number of examples, covered by rule r, from class j, and $N_{i,j}$ be the number of examples where the attribute takes value i in examples of class j.

Assume we observe an example of class c where the value of attribute i is v. The univariate anomaliness score for this attribute is computed as:

$$Uscore_i = 1 - \frac{N_{v,\cdot}}{N} \tag{3}$$

The supervised anomaly score is computed as:

$$U^s score_i = 1 - \frac{N_{v,c}}{N_{\cdot,c}} \tag{4}$$

Again, if $Uscore_i > \lambda$, the attribute value is said to be an anomaly for the context provided by rule r.

Computing the Anomaly Score for Continuous Attributes. For continuous attributes, the statistics stored in $\mathcal{L}_r$ include the mean and standard deviation of each attribute given the class. Remember that these statistics are computed from the examples covered by the rule. Using these statistics we can compute Equation 1 (or Equation 2) using different strategies, including Normal distribution, Z scores, etc. From a set of experiments not described here, the Chebyshev inequality seems to be more effective.

The Chebyshev inequality guarantees that in any probability distribution, 'nearly all' values are close to the mean. More precisely no more than $\frac{1}{k^2}$ of the distribution's values can be more than k standard deviations away from the mean. Although conservative, the inequality can be applied to completely arbitrary distributions (unknown except for mean and variance). Let x be a random variable with finite expected value $\overline{x}$ and finite non-zero variance σ^2. Then for any real number $k > 0$,

$$\Pr(|x - \overline{x}| \geq k\sigma) \leq \frac{1}{k^2}.$$

Only the case $k > 1$ provides useful information. When $k < 1$ the right-hand side is greater than one, so the inequality becomes vacuous, as the probability of any event cannot be greater than one. When $k = 1$ it just says the probability is less than or equal to one, which is always true.

Therefore, for $k = \frac{|x-\overline{x}|}{\sigma}$ and $k > 1$, the anomaliness score is:

$$Uscore_i = 1 - \frac{1}{(\frac{|x-\overline{x}|}{\sigma})^2} \tag{5}$$

The anomaliness score in Equation 5 can be computed using supervised or unsupervised information depending on computing $\overline{x}$ and σ conditioned to the class or not.

Relatively new rules, that are rules that have not been trained with many examples, would more often tend to report a training example as anomaly. In order to prevent this situation, only rules that were trained with more than m_{min} examples are used in the anomaly detection.

Multivariate Score. For each training example and for each attribute value i, we compute the univariate score, $Uscore_i$ using Equation 1. The join degree of anomaliness, assuming that the attributes are independent, is computed for all the attributes such that $Uscore_i > \lambda$, and is given by:

$$\prod_k Uscore_i$$

where k is the set of anomalous attributes.

By applying logarithms, to avoid numerical instabilities, and normalizing, the degree of anomaliness of an example is given by:

$$Ascore = \frac{\sum_{i=1}^{n} \mathcal{I}(Uscore_i)}{\sum_{i=1}^{n} log(Uscore_i)} \tag{6}$$

where

$$\mathcal{I}(x) = \begin{cases} 0 & \text{if } x < \lambda \\ log(x) & \text{otherwise} \end{cases}$$

Equation 6 takes values in the interval $[0, 1]$, where 0 corresponds to the case that none of the attributes is anomalous, and 1 when all the attributes are anomalous.

3.3 Discussion

The main contribution of this work is the ability of incorporating anomaly detection inside the learning process. The advantages of this integration are two-fold. On one hand, the system reports possibly anomalous values, together with an explanation of why each value seems surprising. This information provides insights to the user about the dynamic of the process generating data. On the other hand, the online identification of anomalous examples prevents us to learn from outliers. This is a crucial task in online learning.

Although we illustrate the usability of anomaly detection coupled with a decision rule learner, the proposed method can be used in classification and regression problems with any VFDT like algorithm. The information required to compute the anomaly score is stored in the consequent of rules and in the leaves of a decision tree. Algorithms like decision trees [6] and regression trees [12] can easily incorporate the techniques presented here. The proposed method does not guarantee to find all the anomalies. Moreover, what is an anomaly might depend in the order that examples arrive. The set of rules learned by AVFDR are stable with respect to the order of examples [8].

4 Experimental Evaluation

4.1 UCI Datasets

In a firs set of experiences, and for sanity check, we run the on-line anomaly detection in two artificial datasets - waveform21 [5] and SEA [17]. In these datasets the algorithm did not find any anomalies, which is the correct behavior. In the SEA dataset, shown in Figure 1, the anomaly score [1] is always around 0.

[1] The y axis in the plots showing the distribution of the anomaliness score is in log scale.

Table 1. Anomaly Detection Summary

	Nr.Anomalies	Prequential error	Error in holdout
Adult			
Normal	-	17.58	17.51
Unsupervised	9	17.57	17.51
Supervised	9	17.57	17.51
Covertype			
Normal	-	24.92	38.46
Unsupervised	57	23.75	32.55
Supervised	37	23.91	36.88
Electricity			
Normal	-	18.77	
Unsupervised	189	18.51	
Supervised	13	18.96	
KDDCup99			
Normal	-	0.86	
Unsupervised	10	0.84	
Supervised	17	0.82	

4.2 Real-World Data

The second set of experiments uses well-known datasets were we find anomalies. A summary of the number of anomalies detected is presented in Table 1. For each dataset, we report 3 lines. The first line reports the behavior of AVFDR without detecting anomalies. The second and third line summarizes the behavior of the system using unsupervised and supervised anomaly detection, respectively. The anomalies detected are not used for training the rule learner. The details about these experiments are reported in the following subsections.

Intrusion Dataset. The **KDDCUP 99** is a data set [5] of TCP/IP connections which are labeled either as normal or one of many different types of attacks. In many cases, the attacks are grouped into four categories: DOS (denial-of-service), R2L (unauthorized access from a remote machine), U2R (unauthorized access to local superuser privileges), and probing (surveillance and other probing).

- DOS: denial-of-service, e.g., syn flood;
- R2L: unauthorized access from a remote machine, e.g. guessing password;
- U2R: unauthorized access to local superuser (root) privileges, e.g., various 'buffer overflow' attacks;
- probing: surveillance and other probing, e.g., port scanning.

The test data is not from the same distribution as the training data and moreover there are new attack types that are not in the training data. These new types can be grouped to the categories above as well. The set consist of 4,898,431 and 311,029 instances for training and test respectively.

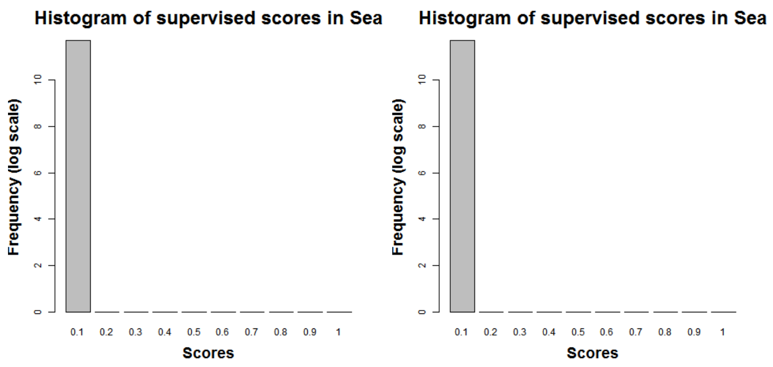


Fig. 1. Distribution of the anomaliness score (supervised and unsupervised) in the SEA dataset

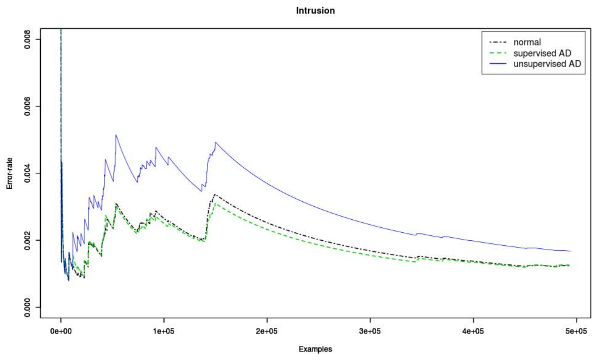


Fig. 2. Anomaly detection influence on prequential error in Intrusion dataset

An example of supervised anomalies found in intrusion dataset:
case 148160:
`dst_host_count=15 [252.2±21.3 class=normal]`
`dst_host_srv_count=13 [248.1±20.9 class=normal]`
in rule:
`count` $\leq 5 \wedge$ `service = private` $\rightarrow$ `Probing`

An example of unsupervised anomalies found in intrusion dataset:
case 121735:
`dst_host_srv_count=163 [254.9±4.51]`
`dst_host_same_srv_rate=0.64 [1.0±0.02]`
in rule:
`count` $> 508 \wedge$ `service = ecr_i` $\rightarrow$ `DoS`

Distribution of the anomaliness score (supervised and unsupervised) in the Intrusion dataset is provided in 3.

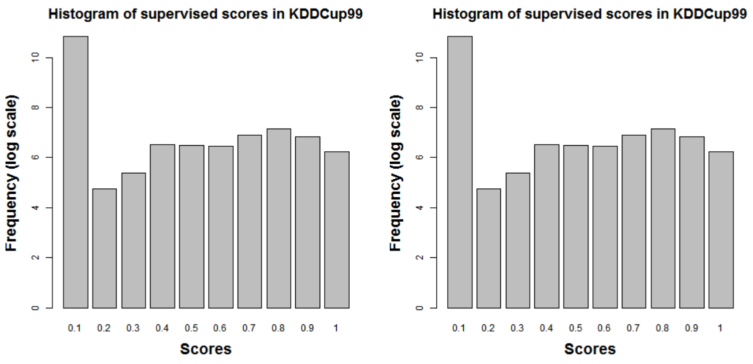


Fig. 3. Distribution of the anomaliness score (supervised and unsupervised) in the Intrusion dataset

Adult Dataset. Data was extracted from the census bureau database in 1994. Prediction task is to determine whether a person makes over 50K a year. The description of data in UCI [5] refers: *A set of reasonably clean records*. The distribution of the anomaliness score (supervised and unsupervised) in the Adult dataset4. Examples of supervised anomalies found in adult dataset:

case 1231
occupation = Priv-house-serv [0 in 216 class=$\leq 50K$]
native-country = France [0 in 216 class=$\leq 50K$]
in rule:
education-num > 12 $\wedge$ marital-status = Never-married $\rightarrow$
$\rightarrow \leq 50K$

case 98361:
occupation = Tech-support [3 in 625 class=$\leq 50K$]
native-country = Peru [0 in 625 class=$\leq 50K$]
in rule:
age $>$ 35 $\wedge$ education-num $<=$ 9 $\wedge$ marital-status = Married-civ-spouse $\rightarrow > 50K$

Electricity Dataset. A widely used dataset is the Electricity Market Dataset introduced in [10]. This time series based data was collected from the Australian New South Wales Electricity Market. The class label identifies the change of the price related to a moving average of the last 24 hours.

Examples of supervised anomalies found in electricity dataset:
case: 7123
day = 6 [0 in 185 class UP]
in rule:
nswprice > 0.102 $\rightarrow$ UP

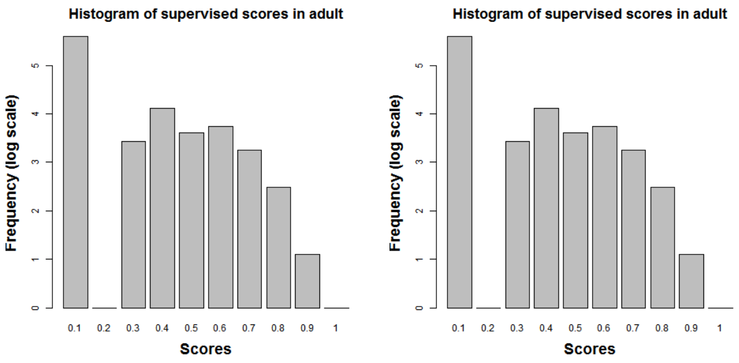


Fig. 4. Distribution of the anomaliness score (supervised and unsupervised) in the Adult dataset

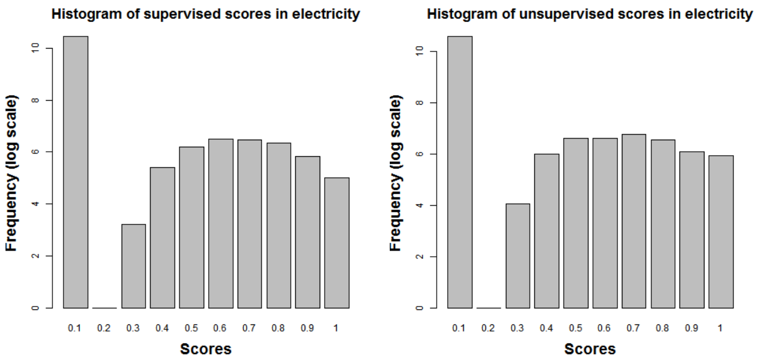


Fig. 5. Distribution of the anomaliness score (supervised and unsupervised) in the Electricity dataset

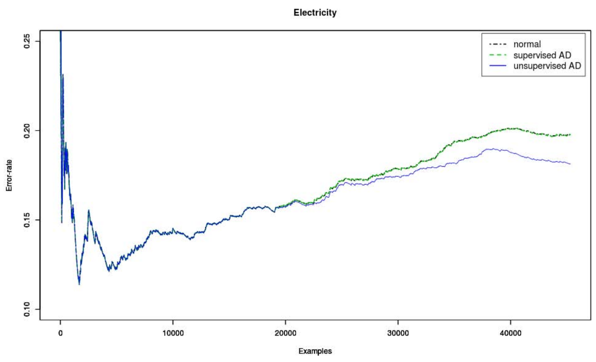


Fig. 6. Anomaly detection influence on prequential error in Electricity dataset

The distribution of the anomaliness score (supervised and unsupervised) in the Electricity dataset is presented in 5. Examples of unsupervised anomalies found in electricity dataset:

case: 17434

```
vicdemand = 0.032626 [0.423±7.15E-8]
transfer = 0.500526 [0.41±6.96E-8]
```

in rule:

`date` $> 0.0131 \wedge$ `nswprice` $\leq 0.0425 \rightarrow$ `UP`

5 Conclusions

In this paper we present a one-pass, streaming algorithm for learning classification rules able to detect contextual anomalies in the data. Contextual anomalies are surprising attribute values in the context defined by the conditional part of the rule. The anomalies detected are characterized by a *context* that refers the region of the instance where the anomaly was detected, and behavioral attributes, those with anomalous values. For each example we compute the degree of anomaliness based on the probability of the attribute-values given the conditional part of the rule covering the example. Our system reports two types of anomalies: supervised and unsupervised anomalies. The examples with high degree of anomaliness are signaled to the user and not used to train the classifier. This is the main claim of this paper: online algorithms benefit from online anomaly detection by rejecting anomalous examples. The experimental evaluation in real-world data sets shows the ability to discover anomalous examples in well-known datasets. The main advantage of the proposed method is the ability to inform the context and explain why the anomaly occurs.

Acknowledgments. We thank the anonymous referees for their constructive comments. The authors acknowledge the financial support given by the project FCT-KDUS PTDC/ EIA/ 098355/2008 FCOMP -01-0124-FEDER-010053, the ERDF through the COMPETE Programme and by National Funds through FCT within the project FCOMP - 01-0124-FEDER-022701. Petr Kosina also acknowledges the support of Fac. of Informatics, MU, Brno.

References

1. Barateiro, J., Galhardas, H.: A survey of data quality tools. Datenbank-Spektrum 14, 15–21 (2005)
2. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. 41(3) (2009)
3. Clark, P., Boswell, R.: Rule induction with cn2: Some recent improvements, pp. 151–163. Springer (1991)
4. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Ramakrishnan, R., Stolfo, S.J., Bayardo, R.J., Parsa, I. (eds.) KDD, pp. 71–80. ACM (2000)
5. Frank, A., Asuncion, A.: UCI machine learning repository (2010)

6. Gama, J., Fernandes, R., Rocha, R.: Decision trees for mining data streams. Intelligent Data Analysis 10, 23–45 (2006)
7. Gama, J., Rocha, R., Medas, P.: Accurate decision trees for mining high-speed data streams. In: Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining. ACM Press, New York (2003)
8. Gama, J., Kosina, P.: Learning decision rules from data streams. In: Walsh, T. (ed.) IJCAI, pp. 1255–1260. IJCAI/AAAI (2011)
9. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2012)
10. Harries, M., Sammut, C., Horn, K.: Extracting hidden context. Machine Learning 32, 101–126 (1998)
11. Hodge, V.J., Austin, J.: A survey of outlier detection methodologies. Artificial Intelligence Review 22(2), 85–126 (2004)
12. Ikonomovska, E., Gama, J., Dzeroski, S.: Learning model trees from evolving data streams. Data Min. Knowl. Discov. 23(1), 128–168 (2011)
13. Knorr, E.M., Ng, R.T., Tucakov, V.: Distance-based outliers: algorithms and applications. The VLDB Journal 8(3-4), 237–253 (2000)
14. Kosina, P., Gama, J.: Handling time changing data with adaptive very fast decision rules. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part I. LNCS, vol. 7523, pp. 827–842. Springer, Heidelberg (2012)
15. Pham, D.-S., Venkatesh, S., Lazarescu, M., Budhaditya, S.: Anomaly detection in large-scale data stream networks. Data Mining and Knowledge Discovery (to appear)
16. Ross Quinlan, J.: Kdd-99 panel on last 10 and next 10 years. SIGKDD Explorations 1(2), 62 (2000)
17. Nick Street, W., Kim, Y.: A streaming ensemble algorithm (sea) for large-scale classification. In: KDD, pp. 377–382 (2001)
18. Tukey, J.W.: Exploratory Data Analysis. Addison-Wesley (1977)
19. Zliobaite, I., Gabrys, B.: Adaptive preprocessing for streaming data. IEEE Transactions on Knowledge and Data Engineering 99(PrePrints), 1 (2012)

Generalizing from Example Clusters

Pan Hu[1,2], Celine Vens[1], Bart Verstrynge[1], and Hendrik Blockeel[1,3]

[1] KU Leuven, Departement of Computer Science,
Celestijnenlaan 200A, 3001 Leuven, Belgium
[2] Ecole des Mines, Saint-Etienne, France
[3] Leiden Institute of Advanced Computer Science,
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

Abstract. We consider the following problem: Given a set of data and one or more examples of clusters, find a clustering of the whole data set that is consistent with the given clusters. This is essentially a semi-supervised clustering problem, but it differs from previously studied semi-supervised clustering settings in significant ways. Earlier work has shown that none of the existing methods for semi-supervised clustering handle this problem well. We identify two reasons for this, which are related to the default metric learning methods not working well in this situation, and to overfitting behavior. We investigate the latter in more detail and propose a new method that explicitly guards against overfitting. Experimental results confirm that the new method generalizes much better. Several other problems identified here remain open.

Keywords: Clustering, Semi-supervised Clustering, Constraint-based Clustering, Metric Learning.

1 Introduction

The task of clustering data is ubiquitous in knowledge discovery. Partitional (or non-hierarchical) clustering can be defined as the following task: given a dataset D, partition D into subsets ("clusters") such that instances within the same cluster tend to be similar, and instances in different clusters dissimilar. The notion of "similarity" is crucial here: depending on how this is defined, different solutions will be found. This is true especially for high-dimensional spaces, where different subspaces may reveal different clusterings [1].

It is not always easy for a user to define a good similarity measure. However, users may be able to give examples of instances that in their opinion should, or should not, belong to the same cluster. The clustering system may use this information to understand better the notion of similarity that the user has in mind, and as a consequence produce a better clustering. This type of clustering setting is called semi-supervised clustering, or constraint-based clustering, as the user gives partial information about the desired clustering in the form of constraints that the clustering must fulfill.

Most existing methods for semi-supervised clustering allow the user to provide a number of so-called *must-link* and *cannot-link* constraints, indicating for pairs of instances whether they should (not) be in the same cluster. Vens et al. [11] recently introduced a slightly different setting, called "semi-supervised clustering with example clusters". The task can be formulated as follows: given a set of data and one or more examples of clusters in the data, find a clustering of the entire data set that is consistent with these example clusters. While this task could in principle be formulated for hierarchical as well as partitional clustering, we focus here on partitional clustering, as do Vens et al.

This type of supervision is often quite natural. Take, for instance, entity resolution in a database of authors: the task is to cluster occurrences of author names on papers such that occurrences are in the same cluster if they refer to the same actual person.[1] If one person indicates all the papers she authored, that set of papers is an example cluster. Knowing one, or a few, such clusters may help the system determine what kinds of clusters are good, so it can better cluster the other instances. Similarly, when clustering images of faces, one may want to cluster according to identity, poses, emotions, etc. Providing a few example clusters may be easy, and may help the system optimize its similarity measure.

Strictly speaking, the task of clustering from example clusters is a special case of semi-supervised clustering using must-link and cannot-link constraints. Indeed, an example cluster can be translated to a set of such constraints. There are disadvantages to such a translation, however: it can generate a large number of pairwise constraints, and these will be distributed very unevenly over the data set. This is a rather extreme setting for standard semi-supervised clustering methods, and it is not obvious that existing methods can handle it well. In fact, Vens et al. show experimentally that they do not, and also show that a method that explicitly addresses the problem can do better. They do not provide much insight into why this is, though.

In this paper, we analyze the problem of semi-supervised clustering with example clusters in more detail. We provide insight into why existing methods for semi-supervised clustering do not handle this type of problem very well. We identify two reasons. First, most of these methods learn a Mahalanobis distance metric that is more consistent with the given constraints; an alternative view on this is that they implicitly transform the data in such a way that data points that should end up in the same cluster are drawn closer together (and data points that should not, are not). We illustrate visually that these methods may not have the desired effect when the pairwise constraints are concentrated in one cluster. Second, the learned metric has a tendency to overfit the example clusters, especially when learning from few and/or small example clusters, and when learning in high-dimensional spaces. This overfitting was mentioned by Vens et al., but not studied in detail, and no solution was proposed. We here propose an improvement to Vens et al.'s method that explicitly takes the overfitting into account and guards against it. We show that the improved method yields considerably better results.

The remainder of this paper is structured as follows. We first briefly survey the work on semi-supervised clustering (Section 2), including the recent work by Vens et al. [11], on which we build. We next identify two problems that these methods suffer from, and investigate empirically to what extent they do (Section 3). These observations lead to an improved version of Vens et al.'s method, which we describe and empirically evaluate in Section 4. Section 5 presents our conclusions.

[1] This task is not trivial because different persons may have the same name, and the same person may be referred to in different ways, e.g., "John Smith", "J.L. Smith".

2 Semi-supervised Clustering

2.1 Using Pairwise Constraints

Most research on semi-supervised clustering has focused on providing pairwise constraints to the clustering algorithm. In their seminal paper, Wagstaff et al. [12,13] introduce the concept of must-link and cannot-link constraints for specifying, respectively, that two instances should, or should not, be in the same cluster.

One way of dealing with these pairwise constraints is adapting existing clustering algorithms to take them into account. Wagstaff et al. [13] adapt K-Means to this effect. The resulting Constrained Kmeans algorithm iteratively updates cluster centers and cluster assignments exactly like K-Means does, but with the exception that an instance cannot be assigned to a cluster if that would cause constraints to be violated.

Alternatively, one can use a standard algorithm, but adapt the distance metric. Xing et al. [15] propose to learn a Mahalanobis matrix M [8], which defines a corresponding Mahalanobis distance

$$d_M(x,y) = \sqrt{(x-y)^T M(x-y)} \tag{1}$$

They find the M that minimizes the sum of squared distances between instances that must link, under the constraint that $d_M(x,y) \geq 1$ for all x and y that cannot link. M can be restricted to be a diagonal matrix, or can be full. The idea of learning such a distance function is generally referred to as metric-based or similarity-adapting methods [7]. An important point to remember is that the Mahalanobis distance d_M is equivalent to the Euclidean distance in the transformed space obtained by multiplying the data matrix by $M^{1/2}$.

Combining algorithm and similarity adaptation, Bilenko et al. [4] introduced the MPCK-Means algorithm. A first difference with Wagstaff et al. is that constraints are now handled in a soft-constrained manner by defining costs for unsatisfied constraints. Furthermore, the k means are initialized using a seeding procedure proposed by Basu et al. [3]. For metric learning, MPCK-Means provides two options: a separate Mahalanobis metric can be learned for each tentative cluster in every iteration of the algorithm, allowing clusters of different shapes in the final partition (we refer to this as local metric learning), or a single Mahalanobis metric can be learned for the whole space (global metric learning).

2.2 Using Chunklets

Alternatively to pairwise constraints, Bar Hillel et al. [2] use *chunklets*, groups of instances that are known to belong to the same cluster. Their Relevant Component Analysis (RCA) algorithm takes chunklets as input and learns a Mahalanobis matrix. This approach is shown to work better than Xing et al.'s for high-dimensional data. A downside is that only must-link information is taken into account. There is no information about which instances cannot link: different chunklets may belong to the same cluster, or they may not. RCA minimizes the same function as Xing et al.'s method, but under different constraints [2].

Yeung and Chang [16] have extended RCA to include cannot-link information. They treat each pairwise constraint as a chunklet, and compute a separate matrix for the must-link constraints, A_{ML}, and for the cannot-link constraints, A_{CL}. The data are then transformed by $A_{CL}^{1/2} \cdot A_{ML}^{-1/2}$. This "pushes apart" cannot-link instances in the same way that must-link instances are drawn together. It is equivalent to learning a Mahalanobis distance metric d_M with $M = A_{ML}^{-1/2} \cdot A_{CL} \cdot A_{ML}^{-1/2}$.

2.3 Using Example Clusters

Vens et al. [11] formally define the task of semi-supervised clustering with example clusters as follows ($\mathcal{P}(\cdots)$ denotes the power set):

Given: An instance space X, a set of instances $D \subseteq X$, a set of disjoint example clusters $E \subseteq \mathcal{P}(D)$, and a quality measure $Q : \mathcal{P}(D) \times \mathcal{P}(D) \rightarrow \mathbb{R}$.
Find: A partition $\mathbf{C} = \{C_1, C_2, ..., C_k\}$ over D that maximizes $Q(\mathbf{C}, E)$.

Q typically measures to what extent $\mathbf{C}$ is consistent with E (ideally, $E \subseteq \mathbf{C}$), but may also take general clustering quality into account. The number of clusters to be found, or the distance metric to be used, are not part of the input. The requirement that $E \subseteq \mathbf{C}$ is not strict; this allows for noise in the data.

Applications for this setting include all problems where it is easy to provide reasonably accurate example clusters (e.g., entity resolution in author databases, face recognition in picture databases), and where different natural clusterings may exist (e.g., clustering in high-dimensional spaces, where different subspaces may reveal different clusterings, or clustering data that exhibit structure at different levels of granularity).

Example clusters can easily be translated into pairwise constraints, but this results in many constraints (for a single example cluster with n instances in a dataset with N instances, this number is $O(nN)$), and these are highly concentrated (they all involve the example cluster) [11]. This poses problems for several of the existing methods, as we shall see further on. Alternatively, cluster examples can be seen as a special type of chunklets, namely, "maximal" chunklets. RCA can therefore be applied without any translation, but then the information about the maximality of the chunklets is lost (RCA allows separate chunklets to end up in the same cluster, which is not wanted when chunklets are known to be maximal). Yeung and Chang's extension allows for negative information, which solves this problem; but this negative information is again expressed by means of pairwise constraints. Thus, although the chunklet based methods provide a concise representation for the must-link constraints, they do not provide one for cannot-link constraints, so they, too, suffer from the problem of generating many constraints.

Motivated by the above considerations, Vens et al. propose the CLUE algorithm. As we build upon this algorithm, we next discuss it in some more detail.

2.4 The CLUE Algorithm

Algorithm 1 [11] presents CLUE. This algorithm uses a dataset and one or more example clusters as inputs. It does not require the user to provide the number of clusters or a particular distance metric.

Algorithm 1. CLUE

```
Input: $D$: a data set, $E$: a set of example clusters $\{E_i\}_{i=1}^k$ with $E_i \subseteq D$
Output: a partition $P$ of $D$
Algorithm:
1. Rescale all attributes linearly to [0,1]
2. Learn a Mahalanobis distance $d_M$ that is adapted to the constraints
3. Construct a dendrogram $\Delta$ by applying a bottom-up hierarchical clustering procedure with $d_M$
4. Find the range of partitions in $\Delta$ for which the examples clusters are reconstructed optimally
5. Within that range, find the best partition $P$
```

The algorithm computes a distance metric using an approach similar to that of Yeung and Chang [16]. It computes a Mahalanobis matrix $M = A_{ML}^{-1/2} \cdot A_{CL} \cdot A_{ML}^{-1/2}$, with

$$A_{ML} = \frac{1}{N_a} \sum_{E_i \in E} \sum_{x \in E_i} (x - \bar{E}_i)(x - \bar{E}_i)^T \tag{2}$$

$$A_{CL} = \frac{1}{N_b} \sum_{E_i \in E} \sum_{x \notin E_i} (x - \bar{E}_i)(x - \bar{E}_i)^T \tag{3}$$

where $N_a = \sum_{E_i \in E} |E_i|$ and $N_b = |D| \cdot |E| - \sum_{E_i \in E} |E_i|$.

Thus, while Yeung and Chang use as chunklets the pairwise constraints, CLUE uses as "positive" chunklets the example clusters, and as negative chunklets, pairs $(x, \bar{E}_i)$ with $x \notin E_i$ and $\bar{E}_i$ the mean of the cluster. Intuitively, this pulls instances in the example closer to its center, and pushes other instances farther away from it. The method has complexity $O(n + kN)$, with k the number of example clusters, n the total number of instances in these, and N the overall number of instances, as opposed to $O(nN)$ if Yeung and Chang's method were used.

Next, a standard bottom-up hierarchical clustering method is used, using d_M as distance metric, and using single or complete linkage. In this paper we use the latter. This gives a dendrogram that represents N partitional clusterings, from N singletons at the bottom to a single cluster at the top. This dendrogram is then cut as follows. At each level, the corresponding partition is evaluated using the so-called CORI measure, which tests how well this partition reconstructs the example cluster. With S_{ML} and S_{CL} the set of, respectively, all must-link and cannot-link constraints induced by the example clusters, and $S_{ML}^{correct}$ and $S_{CL}^{correct}$ the sets of all those constraints fulfilled by a clustering C, the CORI of C is defined as:

$$\text{CORI}(C) = \left(\frac{|S_{ML}^{correct}|}{|S_{ML}|} + \frac{|S_{CL}^{correct}|}{|S_{CL}|} \right) /2 \tag{4}$$

The CORI measure equals 0.5 at the lowest and upper level, and reaches 1 when all example clusters occur in the clustering (i.e., are identical to one C_i). Note that the must-link and cannot-link components of the CORI each have a total weight of 0.5, regardless of how many constraints of each type there are. This is because usually $|S_{CL}| \gg |S_{ML}|$.

The returned clustering is the one with highest CORI. When multiple clusterings have the same highest CORI, the one with best overall clustering quality is chosen, as measured by category utility [5]. This is an evaluation metric that judges cluster quality in terms of intra- and inter-cluster dissimilarity. Witten et al. [14] proposed a version of category utility for numeric data. CLUE uses "weighted category utility" (WCU), a variant of Witten et al.'s version that takes the Mahalanobis distance into account; it corresponds to implicitly transforming the data according to $M^{1/2}$ and then computing category utility exactly as proposed by Witten et al.

3 Analysis

Our analysis of the behavior of semi-supervised clustering algorithms when learning from example clusters is largely empirical. We first describe the datasets used in the analysis, then the evaluation measures, then we consecutively discuss two weaknesses that the current methods suffer from. In most of this analysis, we assume only one example cluster is given (this is in a sense the most extreme case). The CLUE algorithm is consistently run with Complete Linkage hierarchical clustering [9].

3.1 Datasets

We use the same datasets as in Vens et al. [11]. One is a **synthetic dataset** with 200 instances and 6 numeric dimensions with different domain sizes, see Figure 1. Three dimensions were randomly generated, one dimension contains five bar-shaped clusters, and two dimensions together form 16 circle-shaped clusters. Thus, two possible target clusterings are embedded in these data, which we refer to as **Bars** and **Circles**.

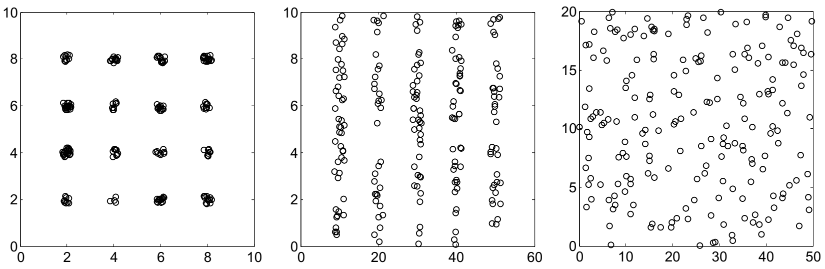


Fig. 1. The 6 dimensions of the synthetic dataset

Besides this, three UCI datasets [6] are used. **CMU Face Images** contains 640 pictures of 20 different persons[2], each shown with 4 poses, 4 emotions, and with or without sunglasses. This is an example of a dataset that can naturally be clustered in several ways. We use the "identity" and "pose" as target clusterings, leading to the **Identity** and **Pose** tasks. Principal Component Analysis was applied to the original data to represent the images as linear combinations of eigenfaces [10]. Only the first 100 eigenfaces were kept; this allowed us to represent the data in a more compact way, while preserving 97% of the original variance in the data. **Libras Movement** contains 15 classes

[2] Due to a corrupted image file, the identity "karyadi" was left out, resulting in 19 identities.

of 24 instances each. Each class references to a hand movement type in Brazilian signal language. **Seeds** contains measurements of seven geometrical properties of kernels belonging to three different varieties of wheat. It has 210 instances.

3.2 Evaluating Clusterings

We use several cluster evaluation measures. In the following, $P = \{p_1, p_2, ..., p_k\}$ denotes the predicted clustering, and $T = \{t_1, t_2, ..., t_l\}$ the target clustering. $c(x)$ denotes the event that an instance x is in a particular cluster c, and $Pr(c(x))$ the probability of this event if x is selected randomly from the data (with c typically of the form p_i or t_j).

The **Rand Index** (RI) is commonly used to compare a predicted clustering to a target clustering. It expresses the proportion of instance pairs for which both clusterings agree on whether they are in the same cluster or not.

When there are many clusters, RI can be dominated by instances correctly predicted not to be in the same cluster. For instance, if $P = \{\{a,b\}, \{c,d\}, \{e,f\}, \{g,h\}, \{i,j\}\}$ and $T = \{\{j,a\}, \{b,c\}, \{d,e\}, \{f,g\}, \{h,i\}\}$, we obtain $RI = \frac{0+35}{45} = 0.778$, a high score for a bad clustering. The **Weighted Rand Index** (WRI) cancels this effect by giving the same importance to the set of pairs that should be in the same clusters, and the set of pairs that should not be (each get weight 0.5). For the clustering given above, we obtain $WRI = \frac{1}{2} \cdot \frac{0}{5} + \frac{1}{2} \cdot \frac{35}{40} = 0.438$

Normalized mutual information (NMI) [9] measures the amount of information that is shared by two clusterings, and penalizes large clusterings. It is defined as follows:

$$NMI(P,T) = \frac{MI(P,T)}{(H(P) + H(T))/2} \tag{5}$$

where MI is mutual information:

$$MI(P,T) = \sum_{i=1}^{k} \sum_{j=1}^{l} Pr(p_i(x), t_j(x)) \cdot \log \left(\frac{Pr(p_i(x), t_j(x))}{Pr(p_i(x)) \cdot Pr(t_j(x))} \right)$$

and H is entropy:

$$H(\{c_1, c_2, \ldots, c_n\}) = -\sum_{i=1}^{n} Pr(c_i(x)) \cdot \log (Pr(c_i(x)))$$

This measure gives a higher weight to larger clusters, which can be undesirable when cluster sizes may differ substantially. For instance, consider $T = \{\{a\}, \{b\}, \{c\}, \{d,e,f\}\}$ and $P = \{\{a,b,c\}, \{d,e,f\}\}$. If we add more and more instances to the last cluster in both T and P, then $H(P)$ and $H(T)$ will get closer to zero, making P a better clustering, although it still correctly identifies only one of the four clusters in T.

To deal with this, Vens et al. propose a new clustering evaluation measure, called **complemented entropy** (CE). It scores the entropy of the target labels in the predicted clusters (H_t), as well as the entropy of the predicted labels in the target clusters (H_p). These entropies are in a sense complementary: predicting too few clusters increases H_t, predicting too many increases H_p. A formal definition is given below:

$$H_t = -\sum_{i=1}^{k} \sum_{j=1}^{l} Pr(t_j(x) \mid p_i(x)) \cdot \log (Pr(t_j(x) \mid p_i(x)))$$

$$H_p = -\sum_{j=1}^{l}\sum_{i=1}^{k} Pr(p_i(x) \mid t_j(x)) \cdot \log\left(Pr(p_i(x) \mid t_j(x))\right)$$

$$CE = 1 - \left(\frac{H_t}{maxH_t} + \frac{H_p}{maxH_p}\right)/2 \qquad (6)$$

In this definition, $maxH_t$ denotes the highest possible value for H_t, and is reached when all predicted clusters contain an equal number of target labels, and similarly for $maxH_p$.

Whereas NMI gives more weight to clusters with high cardinality, CE gives equal weight to each cluster, irrespective of its cardinality. In the previous example, if we increase the number of instances in the last clusters, CE remains unchanged.

3.3 The Mahalanobis Metric

In this and the following section, we discuss two problems that the current methods for metric learning suffer from, when used with example clusters. First, we show that the learned Mahalanobis metric will not always transform the data in a good way. Second, we show that virtually all metric learning methods overfit the example cluster.

Consider the dataset shown in Figure 2, consisting of four square-like clusters (points drawn from a two-dimensional uniform distribution). Figure 3 shows the transformed instances for two methods that learn a Mahalanobis metric: the method by Xing et al. [15] and the CLUE method [11].

Although the clusters in the original dataset are well-separated, both methods transform the data significantly. By skewing the original clusters, the transformations actually often lead to a worse configuration for clustering, rather than a better one. While in the original dataset, points belonging to different clusters are rarely closer to each other than points drawn from the same cluster, this happens much more frequently in the transformed dataset. Furthermore, the skew that is introduced depends on the example cluster: using a different example cluster results in a very different transformation.

To understand what happens, observe that metric learners tend to push together pairs that must link, and draw apart pairs that cannot link (in the case of Xing et al., these are not drawn apart but a lower bound on their distance is imposed). Assume the lower left cluster is used as an example; then must-link constraints cause compression of the space in mostly random directions, but cannot-link constraints cause expansion of the space mostly to the right, up, or upper right (on average around a line with slope 1). The same holds when the example cluster is the upper right one. For the other two clusters, expansion happens on average around a line with slope -1. This is clearly visible in the figures. Although Xing et al.'s method does not expand the space in this directions, it avoids compression in exactly the same direction, while compressing maximally in the perpendicular direction, so a similar effect is obtained. The one run where no compression is obtained, is where compression is not possible without violating constraints (i.e., the points cannot be projected onto a line without some clusters overlapping).

The effect is demonstrated here for only two methods, but the other methods exhibit similar behavior. The high concentration of must-link and cannot-link constraints in a small area has the effect of stretching the space in the direction where many other points happen to be. There is no reason to believe that this would yield a good metric, unless

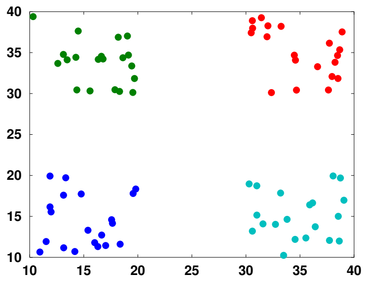


Fig. 2. Synthetic dataset with four clusters

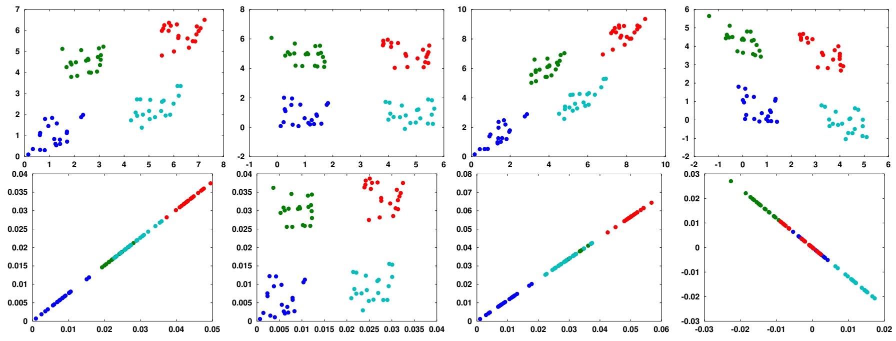


Fig. 3. Transformed data for the synthetic dataset. Metric learning methods: CLUE (row 1) and Xing et al. (row 2). Example clusters: bottom left cluster (column 1), top left cluster (column 2), top right cluster (column 3), bottom right cluster (column 4).

one only wants to separate the one example cluster from all the others. On the contrary, it can have a detrimental effect, as shown here.

3.4 Overfitting

In their earlier experiments with CLUE, Vens et al. observe that it returns too many clusters. It turns out that the CORI measure peaks too early. This can only be explained if example clusters are reconstructed much earlier than other clusters, which means the metric is overfitting the example clusters in the sense that pairs of instances in these clusters are drawn together much more than pairs in other clusters.

That this is indeed the case, is clearly visible in Figure 4, which shows the dendrogram constructed by CLUE for the Libras dataset with a single example cluster. The example cluster is shown in red. It is clear that the metric pushes together the pairs of instances in the example cluster much more than for other clusters, otherwise all clusters should be reconstructed at approximately the same level.

We have measured the degree of overfitting for all methods, as follows. We consider all instance pairs that belong to the same cluster, and sort them according to the learned

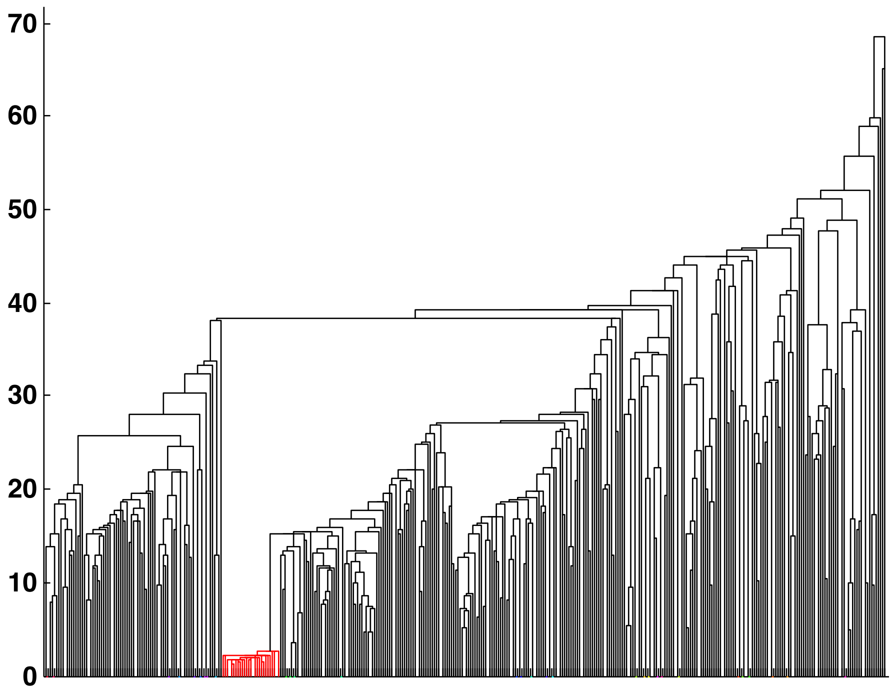


Fig. 4. Dendrogram constructed by CLUE for the Libras dataset. The red part corresponds to the example cluster.

distance between the instances. We then compare the sum of the distance ranks for the instance pairs in the example cluster to the average sum of the ranks over all clusters:

$$\frac{\sum_{x_k, x_l \in E} rank(x_k, x_l)}{\sum_{x_k, x_l \in C} rank(x_k, x_l)/|C|}$$

where E denotes the example cluster, and C denotes any cluster. Table 1 shows the overfitting results for all metric learning methods. We have used each cluster as an example cluster once, and report the average results.

The table clearly shows that, apart from MPCK-Means with global metric learning, all methods have a high risk of strongly overfitting the example cluster. One would expect this risk to be higher as the dimensionality of the space increases (more parameters to be tuned) and the size of the example cluster decreases (fewer independent data to tune these parameters). The table confirms this; for instance, overfitting is worse for Identity (19 small clusters) than for Pose (4 large clusters), which have the same dimensionality, and is worse for Libras (90 dimensions) than for Circles (6 dimensions).

MPCK-Means does not overfit the example cluster, because it does not seek the minimization of distances within the example cluster. Instead, it uses an objective function similar to that of K-Means, but adds penalties for violated constraints, and adapts its metric after each iteration. Thus, metric learning and clustering are interleaved, in

Table 1. Overfitting results. The closer the values are to 1, the less overfitting occurs.

Method	Bars	Circles	Identity	Pose	Libras	Seeds
MPCK-MEANS_global	1.008	0.970	0.996	1.001	0.984	1.000
MPCK-MEANS_local	1.188	1.247	0.725	0.547	0.427	0.939
Xing et al.	0.921	0.674	0.068	0.261	0.150	0.831
RCA	0.869	0.590	0.053	0.268	0.060	0.575
Yeung&Chang	0.899	0.604	0.053	0.274	0.090	0.911
CLUE	0.861	0.564	0.053	0.266	0.080	0.908

Table 2. Comparing distance ranks within and between clusters

Method	Bars	Circles	Identity	Pose	Libras	Seeds
original	0.184	0.076	0.014	0.398	0.041	0.424
MPCK-MEANS_global	0.111	0.027	0.033	0.480	0.041	0.699
CLUE	0.066	0.006	0.011	0.360	0.041	0.431

contrast to the other methods which first learn a metric based on constraints only, then perform clustering. All together, this puts less weight on satisfying the constraints.

The above might suggest that MPCK-Means learns a metric that may fit the example cluster less well, but fits the overall clustering better. As it turns out, this is not the case. We have evaluated this hypothesis as follows.

We consider all instances outside the example cluster, and compute their pairwise distances. We sort these distances, and compare the sum of the distance ranks for pairs of instances in the same cluster to the sum of distance ranks for pairs of instances in different clusters:

$$\frac{\sum_{x_k,x_l \in C} rank(x_k,x_l)}{\sum_{x_k \in C_i, x_l \in C_j, i \neq j} rank(x_k,x_l)}$$

We call this the "within/between measure". Clearly, a good metric will decrease this ratio. Table 2 shows the results for MPCK-Means (with global metric) and CLUE, before and after transformation with the learned metric. We observe that only in 2 out of 6 cases, MPCK-Means improves the rank ratios. For Seeds, Identity and Pose, the results are much worse than without metric learning. CLUE, on the other hand, brings improvements in 4 cases, and has no or a slight opposite effect on 2 cases.

We conclude that MPCK-Means indeed avoids overfitting, yet does not successfully adapt its metric to the clustering problem, whereas the other metric learning methods adapt the metric too much, overfitting the example cluster.

4 CLUE with Defense against Overfitting: CLUEDO

The observation that most systems tend to overfit, and that this is likely due to the pairwise constraints being concentrated in part of the input space, points to an opportunity to improve CLUE. The basic reasoning is as follows. The original CLUE system tends to return too many clusters because the CORI reaches its optimal point too early, and this happens because overfitting causes the example cluster to be compressed more strongly than the other clusters. Even though the example cluster gets reconstructed early, some pairs from other clusters have already been merged at this point. It seems relatively safe to assume that most of these pairs indeed belong to the same cluster. One could add such merged pairs to the list of must-link constraints and re-run the clustering process.

We have tried several variants of this idea, for the case of a single example cluster.

1. Run CLUE, find the optimal clustering level, add all pairs of examples that got merged before this level to the list of must-link constraints, repeat until some stopping criterion is fulfilled.
2. Run CLUE, find the lowest clustering level for which the example cluster is reconstructed, add all pairs of examples that got merged before this level to the list of must-link constraints, repeat until some stopping criterion is fulfilled.

It turns out both methods suffer from too many erroneous must-link constraints being added, which results in decreasing, rather than increasing, performance. We therefore decided to add must-link constraints more cautiously. This results in the CLUEDO algorithm, shown as Algorithm 2.

Algorithm 2. CLUEDO

```
Input: $D$: a data set, $E$: a set of example clusters $\{E_i\}_{i=1}^k$ with $E_i \subseteq D$
Output: a partition $P$ of $D$
Algorithm:
1. Rescale all attributes linearly to [0,1]
2. For $i = 1$ to 10 do:
    2.1 Learn a Mahalanobis distance $d_M$ that is adapted to the current constraints
    2.2 Construct a dendrogram $\Delta$ with a bottom-up clustering procedure using $d_M$
    2.3 For all instance sets merged below level $iN/10$ : add pairwise constraints
        for all pairs in these sets, unless they violate the original constraints
3. Find the range of partitions in $\Delta$ for which the examples clusters are reconstructed optimally
4. Within that range, find the best partition $P$
```

Table 3. Overview of CLUEDO results (improvements over CLUE shown in boldface)

		Bars	Circles	Identity	Pose	Libras	Seeds
# Clusters	CLUE	5	22.1	131.5	169.5	254.8	4.3
	CLUEDO	5	**14.9**	**32.1**	**35**	**13.9**	**3**
	true	5	16	19	4	15	3
Overfitting	CLUEDO	**0.89**	**0.97**	**0.70**	**0.38**	**0.69**	**0.95**
Within/between	CLUEDO	0.0664	**0.0054**	**0.0053**	0.3879	**0.0311**	**0.4155**

Table 4. Various evaluation measures for various methods. The last column shows the average rank of the methods. Best results are shown in boldface.

Method	Bars	Circles	Identity	Pose	Libras	Seeds	Avg. rank
			Normalized mutual information (NMI)				
K-MEANS	0.039	0.501	0.717	0.039	0.559	0.641	8.000
CONSTRAINED K-MEANS (CKM)	0.275	0.494	0.738	0.191	0.564	0.747	7.167
MPCK-MEANS_global	0.757	0.281	0.797	0.035	0.535	0.750	6.667
MPCK-MEANS_local	0.391	0.252	0.673	0.026	0.433	0.748	9.000
Xing et al. + CKM	0.889	0.938	0.793	**0.563**	0.548	0.788	3.833
RCA + CKM	0.542	0.631	0.805	0.544	0.580	0.827	4.333
Yeung&Chang + CKM	0.596	0.789	0.775	0.533	0.590	0.879	4.167
CLUE	**1.000**	**0.952**	0.706	0.357	**0.645**	0.453	4.530
CLUEDO	**1.000**	0.934	**0.919**	0.335	0.555	0.660	4.333
CLUEDO + CKM	0.674	0.781	0.885	0.552	**0.645**	**0.883**	**2.667**
			Complemented entropy (CE)				
K-MEANS	0.051	0.529	0.768	0.133	0.615	0.704	8.167
CONSTRAINED K-MEANS (CKM)	0.284	0.519	0.793	0.282	0.603	0.751	7.500
MPCK-MEANS_global	0.757	0.336	0.829	0.074	0.558	0.760	7.167
MPCK-MEANS_local	0.513	0.419	0.711	0.054	0.534	0.764	8.667
Xing et al. + CKM	0.932	0.951	0.840	0.592	0.587	0.792	4.167
RCA + CKM	0.543	0.661	0.845	0.612	0.643	0.835	4.000
Yeung&Chang + CKM	0.602	0.808	0.817	0.590	0.629	0.883	4.667
CLUE	**1.000**	**0.965**	0.780	**0.619**	**0.744**	0.631	3.667
CLUEDO	**1.000**	0.947	**0.940**	0.588	0.633	0.726	3.833
CLUEDO + CKM	0.683	0.800	0.927	0.609	0.672	**0.885**	**3.000**
			Weighted rand index (WRI)				
K-MEANS	0.504	0.587	0.767	0.507	0.662	0.853	8.500
CONSTRAINED K-MEANS (CKM)	0.598	0.590	0.794	0.564	0.664	0.903	6.500
MPCK-MEANS_global	0.838	0.518	0.829	0.515	0.669	0.899	6.000
MPCK-MEANS_local	0.650	0.526	0.730	0.511	0.623	0.891	8.167
Xing et al. + CKM	0.906	0.948	0.832	**0.748**	0.662	0.882	4.167
RCA + CKM	0.717	0.671	0.839	0.734	0.696	0.913	4.000
Yeung&Chang + CKM	0.742	0.798	0.815	0.733	0.692	0.942	4.167
CLUE	**1.000**	0.961	0.658	0.510	0.517	0.679	7.000
CLUEDO	**1.000**	**0.964**	0.894	0.579	0.688	0.835	3.667
CLUEDO + CKM	0.782	0.789	**0.922**	0.744	**0.715**	**0.946**	**2.500**
			Rand index (RI)				
K-MEANS	0.651	0.896	0.930	0.561	0.899	0.853	7.500
CONSTRAINED K-MEANS (CKM)	0.696	0.895	0.928	0.593	0.900	0.903	6.500
MPCK-MEANS_global	0.878	0.866	0.944	0.555	0.888	0.899	6.833
MPCK-MEANS_local	0.670	0.806	0.924	0.554	0.857	0.891	9.000
Xing et al. + CKM	0.918	0.983	0.945	**0.801**	0.897	0.894	3.833
RCA + CKM	0.819	0.918	0.951	0.766	0.895	0.921	4.833
Yeung&Chang + CKM	0.830	0.948	0.946	0.767	0.909	0.948	3.500
CLUE	**1.000**	**0.984**	0.911	0.675	**0.933**	0.679	4.833
CLUEDO	**1.000**	0.975	**0.978**	0.701	0.849	0.835	4.833
CLUEDO + CKM	0.854	0.945	0.961	0.761	0.919	**0.952**	**3.167**

CLUEDO repeatedly runs steps 2 and 3 of CLUE, each time adding more must-link constraints. It first adds the most certain ones, corresponding to pairs that were among the first 10% to get merged (but excluding those pairs that would violate the original constraints). It relearns the Mahalanobis metric with those pairs added, re-clusters with this new metric, then adds also must-link constraints for pairs that got merged in the lower 20% levels. It continues doing this up to the 90% level, then stops and returns the partition with optimal CORI and WCU, as in CLUE.

Table 3 summarizes the results obtained with CLUEDO. It confirms that the number of clusters returned by CLUEDO is often much lower than that of CLUE; also the overfitting metric improves substantially and consistently. This shows that the method achieves its goal of guarding against the overfitting behavior discussed earlier. The within/between measure improves 4 times out of 6, and worsens 1 time.

To put the results into context, we include in Table 4 the NMI, CE, and (W)RI results for a variety of approaches, most of which consist of running constrained K-means with the right number of clusters, and with a metric learned by one of the alternative metric learning methods. Although the evaluation measures generally agree which are the better or worse performing methods, there are some differences. For instance, for Libras, CLUE is the best performing method according to NMI, CE, and RI; however, it is the worst method according to WRI. This results in a lower average WRI rank for CLUE. The results further show that, overall, CLUE and CLUEDO are among the better-performing methods, even though they do not know the number of clusters in advance. Comparing CLUEDO to CLUE, we always observe an improved WRI, while the RI, NMI and CE are more mixed. For completeness, we also included the results for CLUEDO combined with constrained K-means (and hence, using the number of clusters as input) instead of using agglomerative clustering to obtain the final partition. It turns out that this combination has the best average rank over all methods, for all evaluation measures.

5 Conclusions

Generalizing from example clusters is a relatively novel setting for semi-supervised clustering. It can in principle be dealt with by methods that can handle pairwise must-link / cannot-link constraints, but from this point of view it is a rather extreme setting. We have investigated the behavior of these methods in this context, and identified several effects that cause them to perform badly. One is the high concentration of pairwise constraints in one area of the input space, another is the tendency to overfit example clusters. We have quantified these effects for several clustering methods. These observations have led us to propose a new semi-supervised clustering algorithm for this setting, CLUEDO, which is shown to substantially improve upon its predecessor CLUE in terms of finding a clustering with a reasonable number of clusters, and in terms of learning a distance metric that much less overfits example clusters.

Many questions remain. We have mostly focused on extreme situations (generalizing from one example cluster), and while our experiments use a variety of datasets, none of these are very large or high-dimensional. Further, an evaluation of the setting in practical application settings, such as entity resolution (in textual, visual, audio, graph, ... data), would be interesting. Semi-supervised clustering by generalizing from example clusters remains a largely unexplored area that in our view has much potential.

Acknowledgements. C.V. is a Postdoctoral Fellow of the Research Foundation - Flanders (FWO-Vlaanderen). Work supported by the Research Foundation - Flanders (G.0682.11) and the KU Leuven Research Fund (GOA 13/010).

References

1. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data. Data Mining and Knowledge Discovery 11(1), 5–33 (2005)
2. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning a mahalanobis metric from equivalence constraints. Journal of Machine Learning Research 6, 937–965 (2005)
3. Basu, S., Banerjee, A., Mooney, R.: Semi-supervised clustering by seeding. In: Proceedings of 19th International Conference on Machine Learning (ICML 2002) (2002)
4. Bilenko, M., Basu, S., Mooney, R.: Integrating constraints and metric learning in semi-supervised clustering. In: ICML, pp. 81–88 (2004)
5. Fisher, D.H.: Knowledge acquisition via incremental conceptual clustering. Machine Learning 2(2), 139–172 (1987)
6. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
7. Grira, N., Crucianu, M., Boujemaa, N.: Unsupervised and Semi-supervised Clustering: a Brief Survey. A Review of Machine Learning Techniques for Processing Multimedia Content, Report of the MUSCLE European Network of Excellence, FP6 (2004)
8. Mahalanobis, P.C.: On the generalised distance in statistics. In: Proceedings National Institute of Science, India, pp. 49–55 (1936)
9. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, New York (2008)
10. Turk, M.A., Pentland, A.P.: Face recognition using eigenfaces. In: Proceedings 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 591(1), pp. 586–591 (1991)
11. Vens, C., Verstrynge, B., Blockeel, H.: Semi-supervised clustering with example clusters. In: Proceedings of the 5th International Conference on Knowledge Discovery and Information Retrieval (accepted, 2013)
12. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 1103–1110 (2000)
13. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained K-means clustering with background knowledge. In: ICML, pp. 577–584. Morgan Kaufmann (2001)
14. Witten, I., Frank, E., Hall, M.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann (2011)
15. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. In: Advances in Neural Information Processing Systems, vol. 15, pp. 505–512. MIT Press (2002)
16. Yeung, D., Chang, H.: Extending the relevant component analysis algorithm for metric learning using both positive and negative equivalence constraints. Pattern Recognition 39(5), 1007–1010 (2006)

Clustering Based Active Learning for Evolving Data Streams

Dino Ienco[1], Albert Bifet[2], Indrė Žliobaitė[3], and Bernhard Pfahringer[4]

[1] Irstea, UMR TETIS, Montpellier, France
LIRMM, Montpellier, France
dino.ienco@irstea.fr
[2] Yahoo! Research Barcelona, Catalonia, Spain
abifet@yahoo-inc.com
[3] Aalto University and Helsinki Institute for Information Technology, Finland
indre.zliobaite@aalto.fi
[4] University of Waikato, Hamilton, New Zealand
bernhard@cs.waikato.ac.nz

Abstract. Data labeling is an expensive and time-consuming task. Choosing which labels to use is increasingly becoming important. In the active learning setting, a classifier is trained by asking for labels for only a small fraction of all instances. While many works exist that deal with this issue in non-streaming scenarios, few works exist in the data stream setting. In this paper we propose a new active learning approach for evolving data streams based on a pre-clustering step, for selecting the most informative instances for labeling. We consider a batch incremental setting: when a new batch arrives, first we cluster the examples, and then, we select the best instances to train the learner. The clustering approach allows to cover the whole data space avoiding to oversample examples from only few areas. We compare our method w.r.t. state of the art active learning strategies over real datasets. The results highlight the improvement in performance of our proposal. Experiments on parameter sensitivity are also reported.

1 Introduction

Today, large amounts of data are being generated continuously, and we are creating more data every two days, than all the data we created before 2003 [15]. Data streams pose new serious challenges to the data analysis community. To learn supervised models, we need to obtain true labels from the instances of the streams . This labeling phase is usually an expensive and tedious task for domain experts. Consider, for example, textual news arriving as a data stream. The goal is to predict if a news item will interest a given user at a given time. The interests of the user may change over time. To obtain training data, news need to be labeled as interesting or not interesting. This requires human labor. For instance, Amazon Mechanical Turk[1] provides a marketplace for intelligent human labeling.

[1] https://www.mturk.com

Labeling can also be costly because it requires expensive, intrusive or destructive laboratory test. Consider a production process in a chemical plant where the goal is to predict the quality of production output. The relationship between input and output quality might change over time due to constant manual tuning, complementary ingredients or replacement of physical sensors. In order to know the quality of the output (the true label) a laboratory test needs to be performed which is costly. Under such conditions it may be unreasonable to require true labels for all incoming instances.

A way to alleviate this issue is to ask for labels, over time, for only a small and reasonable portion of the data. The main question then is: How can we select a good subset of instances for learning a model? Such a learning scenario is referred to as *active learning*.

Active learning studies how to label selectively instead of asking for all true labels. It has been extensively studied in pool-based [13] and online settings [5]. In pool-based settings the decision concerning which instances to label is made by ranking all historical instances (e.g. according to uncertainty) while in online active learning each incoming instance is compared to a threshold (e.g. an uncertainty threshold) and the system asks for the true label if the threshold is exceeded. The main difference between online active learning and active learning in data streams is in expectations around changes. In data streams the relationship between the input data and the label may change (concept drift) and these changes can happen anywhere in the instance space while online learning assumes a stationary relationship between examples and their labels. As mentioned before, concept drifts in streams can happen anywhere in the data.

To cope with this issue previous works exploit randomization strategies to span the whole instance space [24]. We propose a clustering based approach *ACLStream* (**A**ctive **C**lustering **L**earning for Data **Stream**s)to better deal with possible drifts. More specifically, when a batch of instances arrives, we first partition these instances using a clustering algorithm. Then we query examples for labeling by combining geometrical information (supplied by the clustering) and the maximum a posteriori probability of the model learnt from all the labelled examples from previous batches of data. Once the query labels are obtained, the data stream classifier is updated and it is ready to classify new incoming data. The proposed strategy allows to selectively sample a subset of well distributed instances in the data space. We demonstrate that the selected examples summarize the stream sufficiently for learning an evolving classification model.

The remainder of this paper is organized as follows. Section 2 briefly explores the state of the art in active learning for data streams and makes some connections with semi-supervised learning in data streams. The proposed methodology is presented in Section 3. In Section 4 we present experimental results for a number of real world datasets and we also supply a sensitivity analysis of the essential parameters of the approach. Finally, Section 5 concludes the study.

2 Related Work

Online active learning has been the subject of a number of studies, where the data distribution is assumed to be static [2, 5, 9, 20]. The goal is to learn one accurate model with minimum labeling effort. In contrast, in the evolving data streams setting, which is the subject of our study, the goal is to continuously update a model over time so that accuracy is maintained as the data distribution is changing. The problem of label availability in evolving data streams has been the subject of several recent studies [6,10,12,17,22,24] that fall into three main groups.

The first group of works uses semi-supervised learning approaches to label some of the unlabeled data automatically [12, 17, 22], which can only work under the assumption that the class conditional distribution does not change (no concept drift). Semi-supervised learning approaches are conceptually different from the active learning approaches, that are the subject of our study, since the former can only handle changes in the input data distribution, changes in the relation between the input data and the target label cannot be spotted without querying an external oracle as is done in active learning.

The second group of works process data in batches implicitly or explicitly assuming that data is stationary within batches [6, 10, 14, 16]. Such approaches require an external mechanism to handle concept drift. Lindstrom et al. [14] use uncertainty sampling to label the most representative instances within each new batch. They do not explicitly detect changes, instead they use a sliding window approach, which discards the oldest instances. Masud et al. [16] use uncertainty sampling within a batch to request labels. In addition, they use the unlabeled instances with their predicted labels for training (semi-supervised learning approach). A few works integrate active learning and change detection [6, 10] in the sense that they first detect change and only if change is detected do they ask for representative true labels using offline active learning strategies designed for stationary data. In this scenario drift handling and active learning can be considered as two mechanisms operating in parallel, but doing so independently. This is the main difference between this scenario and the last one, which combines the two mechanisms more closely together.

Finally, the third group of works use randomization to capture possible changes in the class conditional distribution [4, 23, 24]. Cesa-Bianchi et al [4] develop an online active learning method for a perceptron based on selective sampling using a variable labeling threshold $b/(b+|p|)$, where b is a parameter and p is the prediction of the perceptron. The threshold itself is based on certainty expectations, while the labels are queried at random. This mechanism could allow adaptation to changes, although they did not explicitly consider concept drift. Zhu et al. [23] build a classifier on a small portion of data within a batch at random and use uncertainty sampling to label more instances within this batch. A new classifier in each batch is needed to take into account concept drift. Zliobaite et al. [24] operate in the pure online setting without batches, where they combine stationary online active learning with randomization over the instance space. Our study moves a step further from just employing randomization over the instances to capture a potential concept drift. We use stratified sampling over the instance space instead, where the strata are determined using a clustering mechanism.

The idea of pre-clustering has been considered for active learning in the stationary setting [18]. The selection criterion gives priority to two types of samples: samples close to the classification boundary and samples which are cluster representatives. This way the prior data distribution can be taken into account when making labeling decisions, which has been shown to work well empirically and theoretically for certain data distributions. Employing clustering is conceptually similar to our approach, however, the motivation behind doing that in our approach is different. We mainly aim at tracking concept drift using this mechanism.

3 Setting and Methods

In this section we describe our new algorithm *ACLStream* (**A**ctive **C**lustering **L**earning for Data **Streams**). We suppose that our incoming data stream is divided into batches. Each batch S_t is associated with the arrival time denoted by the index t: $S = \{S_1, S_2, ..., S_n, ...\}$. This scenario is general enough to model arbitrary real-world data streams. Note that even in the case of a fully incremental scenario, we can still build batches by employing a buffer based procedure to collect examples. Given a data stream S and a budget b we want to learn a classifier cl with only b% of the instances in the stream. How to select the instances to query is challenging for any active learning strategy. As we are working in a batch incremental scenario, this means that if the value of b is 0.2 we can select 20% of the labels for each batch S_t. The proposed strategy is based on the clustering of instances in a batch. To select a query point, first we choose a cluster and then we select as query one of the instances belonging to that cluster. The use of clusters helps the selection strategy to sample queries from different, but still reasonably densely populated parts of the instance space. In this way we hope to improve coverage for the different classes of the problem and to overcome possible concept drift that can appear anywhere in the data space. Once the clustering is produced we need to determine (i) which are the most useful clusters to select, and (ii) given a cluster, which are the most useful instances inside this specific cluster. For both cases we will define a ranking for all respective objects, clusters or instances. Thus we implement a two step procedure. As a *Macro Step*, we need to define how to produce a suitable ranking of clusters and then, as a *Micro Step*, we need to define how to rank the instances inside each cluster in order to retrieve the most informative instances for labeling first.

The use of clustering, to select query instances, allows covering the whole data space, albeit in a more focused manner than purely random sampling. Points from all areas of the instance space may be sampled, which is a welcome property in a data stream scenario in which concept drifts can appear anywhere. Contrary to randomized strategies [24], that also try to cover the whole data space, *ACLStream* is guided by the partitioning of the space induced by the clustering algorithm. One of the advantages is higher robustness with regard to outliers, where purely random sampling might waste valuable labeling resources.

3.1 Macro Step: Evaluating the Importance of a Cluster

After clustering of a batch is finished, we evaluate the quality of each partition employing the classifier *cl* which was trained on all previously labeled data. We classify all the instances of the batch S_t without taking into account the clustering solution. After the classification step, we compute, separately for each cluster, a distribution vector w.r.t. the class values. This means that each cluster C_* will be associated with a vector of length equals to the number of classes. We indicate this vector as V_{C_*}. Each cell of the vector contains the number of instances in the cluster, for which the classifier predicts that specific class value. Intuitively, if the vector is balanced – all the class values are equally probable – that cluster covers a more difficult part of the data space to classify in comparison to a cluster with a very skewed distribution, in which there is a clear predominant class value. Starting from this intuition, we introduce a function to quantify the homogeneity of the predicted class distribution in a cluster. In particular, the proposed measure returns values closer to 0 when class values are equally probable while it returns 1 when the cluster is homogeneous w.r.t. the predicted class values. The homogeneity function is defined as follows:

$$homogeneity(C_*) = \frac{\sum_{i=0}^{|V_{C_*}|} \sum_{j=i+1}^{|V_{C_*}|} |V_{C_*}[i] - V_{C_*}[j]|}{size(C_*) * (|V_{C_*}| - 1)} \tag{1}$$

where $size(C_*)$ is the number of instances in the cluster C_*. The *homogeneity* function is bounded between 0 and 1. The numerator evaluates the variation of the prediction among the different class values to evaluate how far is from a completely equiprobable situation. The numerator is then normalized by the extreme case in which we assume that all the examples, in the cluster, are assigned to the same class value.

Clusters are sorted in increasing value of their *homogeneity*, from perfectly balanced to fully homogeneous.

3.2 Micro Step: Instance Certainty Inside a Cluster

In order to quantify the importance or certainty of each instance, inside each cluster, we combine two different factors. One is based on the centrality of the instance w.r.t. its cluster centroid while the second one exploits the classifier *cl*. To compute the centrality of an instance x_i w.r.t. its cluster C_* we simply use the Euclidean distance between x_i and the centroid of C_* ($centroid(C_*)$). We refer to this quantity using the notation $d(x_i, C_*)$ without explicitly indicating the use of $centroid(C_*)$ to simplify the notation. The second factor is computed by the classifier. We use the maximum a posteriori probability of the classifier for the particular instance x_i. The maximum a posteriori probability is the maximum among all the class probabilities predicted for x_i. We indicate the maximum a posteriori using the notation $MAP_{cl}(x_i)$. Given a cluster C_* and an instance x_i belonging to that cluster, the centralized certainty of the instance is supplied by the following formula:

$$certainty(x_i) = MAP_{cl}(x_i) * d(x_i, C_*) \tag{2}$$

Intuitively we want to select instances i) for which the decision of the classifier is less clear and ii) which are good representatives, or exemplars, for the area covered by the cluster. To do this we combine the $MAP_{cl}(x_i)$ that represents point i) and $d(x_i, C_*)$ corresponding to point ii) by multiplication, and name the resulting quantity the *certainty* of x_i. Then, for each cluster, all instances are sorted by increasing values of certainty. Therefore the most uncertain instances will be selected first for labeling.

3.3 *ACLStream* Strategy

Given a batch of instances S_t and a budget b, our solution uses clustering to select a fraction (b) of instances from batch S_t that are deemed the most informative for training a stream classifier. *cl* first classifies the incoming instances, then we cluster the elements in S_t, obtaining a partition $C = \{C_1, ..., C_k\}$. After clustering, we apply our sampling strategy. First we perform the macro-step that ranks the clusters in C according to the *homogeneity* function defined in Equation 1. After that, for each cluster, we perform the micro-step which ranks the instances according to their *certainty* (Equation 2). The returned result is a set X of instances selected for labeling. The procedure is summarized in Algorithm 1.

Our solution can also be adopted to work in a one-by-one classification scenario. Using a buffer that collects batches of instances, the classifier continuously classifies examples until the batch is full. At that point, we can interrupt the classification process, select query points through the active learning strategy, update the classifier and reset the buffer to restart collecting examples.

Algorithm 1. ***ACLStream***(S_t, cl, b, k)

```
Require: $S_t$: batch of instances
Require: $cl$: classifier
Require: $b$: budget
Require: $k$: number of clusters
 1: $X = \emptyset$
 2: $C$ = clustering($S_t$,$k$)
 3: $L_c = rankClusters(C, cl)$ according to homogeneity (eq. 1)
 4: for all $c \in L_c$ do
 5:     $rankExamples(c, cl)$ according to certainty (eq. 2)
 6: end for
 7: for $i = 1 \to b \times size(S_t)$ do
 8:     $X = X \cup$ dequeue($L_c$[i % k])
 9: end for
10: return $X$
```

As a clustering algorithm we employ the standard K-Means algorithm [21]. Any kind of clustering algorithm could be used to perform this step. The use of K-Means is motivated by its time complexity, which is linear w.r.t. the size of the batch, and by the fact that it is generally used as universal baseline in clustering. Future work will investigate the utility of alternative clusterers. Once the partition is available, we rank the clusters w.r.t. their homogeneity (see Formula 1). Clusters at the top of the ranking are the most balanced ones in terms of their class distributions. Therefore they also represent areas of the instance space over which the classifier is less confident. This ranking is stored in a ranked list L_c. Then, for each cluster c of L_c we rank their instances. This ranking is performed employing Formula 2. At this point we have quantified the usefulness of both clusters and instances. To select instances for labelling, we start from the top ranked cluster and we select and remove the first instance. Then, iteratively, we select one instance from every other cluster in order to explicitly cover all the different areas of the data space. If the budget exceeds the number of clusters, we restart to sample instances again starting from the top of the cluster ranking. If the budget is low and the number of instances to sample is smaller than the number of clusters we only consider clusters in the top of L_c.

3.4 General Classification Schema

Algorithm 2 summarizes the general classification schema. The Main loop simulates the streaming process collecting batches of data. Once a batch is collected, the data is classified and then used as input for the proposed active learning strategy that returns the set of selected query points. The active learning strategy is realized through Algorithm 1. To evaluate classifier performance we adopt the prequential schema. The evaluation through the prequential setting involves two steps: i) classify an instance, and ii) use the same instance to train the learner. To implement this strategy, first we test all the instances in the batch with the classifier cl. Second, we select examples for labeling using *ACLStream*. In this way we respect the constraints imposed by our setting. Function $askLabel(x_j)$ simulates the user in the labeling phase. At the end, the classifier cl is trained over the set X of labeled data. The process continues until the end of the data stream is reached.

4 Experiments

In this section we evaluate the performance and the quality of the proposed *ACLStream*. We compare our algorithm with three other methods that are explicitly designed for active learning over data streams. We use the prequential evaluation procedure: each time an instance arrives, we first test the algorithm on it, and then we decide on whether pay the cost for labeling it and subsequently use it as an input for updating the classifier.

The first method is a baseline approach, also used in [24], that randomly chooses examples for labeling. We call this method *Random*. The second method, also proposed in [24], uses a randomized variable uncertainty strategy that combines the randomization with maximum a posteriori probability and an adaptive method to avoid consuming too much of the budget when a consecutive run of

Algorithm 2. Active Learning Process(S, b, k)

```
Require: $S$: stream of instances
Require: $b$: budget
Require: $k$: number of clusters
 1: Init classifier $cl$
 2: while hasMoreInstances($S$) do
 3:    $S_t$ = extractNextBatch($S$)
 4:    for all $x_j \in S_t$ do
 5:       test($cl$,$x_j$)
 6:    end for
 7:    $X$ = $ACLStream(S_t,cl,b,k)$
 8:    for all $x_j \in X$ do
 9:       $y_j$ = askLabel($x_j$)
10:       train($cl$,$x_j$,$y_j$)
11:    end for
12: end while
```

easy instances is encountered. We call this approach *Rand Unc*. The last competitor is an ensemble approach [23] that uses a maximum variance principle. Given a set of classifiers, in a batch incremental scenario, the instances to label are the ones over which the classifiers disagree the most. In the original work, given a batch, the authors propose to first select the instances to label and then classify the remain instances in the batch.

To have a fair comparison with the other approaches we wait until the end of the batch before executing active learning. This way the model will be trained to classify the instances of the next batch. We call this classifier *MVC* (Maximum Variance Classifier). Using this set up i) we ensure that the budget constraints are always respected during the stream process, ii) the comparison is fair w.r.t. our approach and the other two competitors that suppose a full incremental scenario, and iii) we respect prequential learning schema. For all methods a warm-up step is introduced. In detail, the first 500 instances of each dataset are all labeled and used to train the initial model used by the specific approach. Evaluation only starts after this warm-up step. All the methods need a classification algorithm as a base block to perform the classification and to produce the maximum a posteriori probability. For this reason for our approach, for the *Random* and for the *Rand Unc* strategies we use the classifier proposed in [7]. This classifier is able to adapt itself to drift situations: when the accuracy of the classifier begins to decrease a new classifier is built and trained with new incoming instances. For *MVC* we use the C4.5 algorithm [19] as suggested in the original paper. Always following the original paper we use a window size of 1 000 instances for *MVC* as that size obtains best results, while for *ACLStream* we employ a window size of 100 instances. The default number of clusters is 5. As our approach uses a nondeterministic algorithm to perform the clustering, each result for *ACLStream* is averaged over 30 runs. All our experiments are performed using the MOA data stream software suite [3]. MOA is an open source software framework in Java designed specifically for data stream mining.

4.1 Datasets

To evaluate all the algorithms we use five real world "datasets: *Electricity*, *Cover Type*, *Airlines*, *Poker*, *KDD99*. *Electricity* data [8] is a popular benchmark in evaluating streaming classifiers. The task is to predict the rise or fall of electricity prices (demand) in New South Wales (Australia), given recent consumption and prices in the same and neighboring regions. *Cover Type* data [1] is also often used as a benchmark for evaluating stream classifiers. The task is to predict forest cover types or types of vegetation from cartographic variables. Inspired by [11] we constructed an *Airlines* dataset using the raw data from US flight control. The task is to predict whether a given flight will be delayed, given the information of the scheduled departure. The *Poker* dataset represents all the possible combination of cards in one hand with the corresponding score as the class value. This results in a big dataset with more than 800k instances. The last dataset, *KDD99*, is commonly used as a benchmark anomaly detection task but recently it has also been employed as a dataset for testing data stream algorithms [17]. One of the big problems with this dataset is the big amount of redundancy among instances. To solve this problem we use the cleaned version named NSL-KDD[2]. To build the final dataset we join both training and test data. A summary of the datasets' characteristics is reported in Table 1. We observe that this collection of datasets contains both binary and multi-class classification problems, datasets with different numbers of instances (varying between 42k to 829k) and different numbers of features (from 7 to 54).

For analysis purposes, we also introduce one more dataset, named *Cover Type Sorted*, in which the instances of the *Cover Type* dataset are reordered w.r.t. the attribute *elevation*. Due to the nature of the underlying problem, sorting the instances by the *elevation* attribute induces a natural gradual drift on the class distribution, because at higher elevation some types of vegetation disappear while other types of vegetation appear gracefully. We think that this final set of six datasets is a good benchmark for evaluating the performance of our approach, *ACLStream*, w.r.t. state of the art methods.

Table 1. Dataset characteristics

Dataset	n. of Instances	n. of Features	n. of Classes
Airlines	539 383	7	2
Electricity	45 312	8	2
Cover Type	581 012	54	7
Poker	829 201	10	10
KDD99	148 517	41	2

4.2 Analysis of Classification Accuracies

The final accuracy results are reported in Figure 1. In this experiment we evaluate the different methods, over the different datasets, varying the budget

[2] `http://nsl.cs.unb.ca/NSL-KDD/`

percentage. We start with a budget percentage of 0.03 and go up to a percentage of 0.3. Obviously, to evaluate the results we need to take into account both budget size and accuracy. We can observe that, except for *KDD99* dataset, *ACLStream* outperforms the other methods for low budgets, less than 0.15, while for bigger budgets the performances are either better as well, or at least comparable. For the case of *KDD99* the performances of *Random*, *Rand Unc* and *ACLStream* are very close to each other and the difference is smaller than 1.5 percentage points of accuracy. Another useful feature of our new method is its stability. As we can see from the graphs, *ACLStream* always remains stable varying the budget percentage while this is not the case for all the other methods. This behaviour is clearly present for the *Cover Type* dataset. In this case the *Rand Unc* strategy is very unstable and small changes of the budget (from 0.15 to 0.25) induce a big change (ten points of accuracy) in the final performance. Also the *Random* strategy is much more unstable than our proposed method. Another general conclusion we can draw regards *MVC*, which always obtains low accuracy performance when compared to all the other methods. To wrap up all the findings from this experiment, we can claim that *ACLStream* works well for very low budget percentages, which is a very important and desirable feature for any active learning strategy. This is particularly important in the data stream domain, in which data arrives continuously and time-consuming operations, such as data labeling, need to be minimized. On the other hand, due to its stability, *ACLStream* small budget results are very similar to ones obtained with higher budget. This observation implies that the active learning strategy based on our clustering approach is especially effective for small budgets.

4.3 Influence of the Number of Clusters

We evaluate how the number of clusters influences the performance of *ACLStream*. For this analysis we fix the batch size to 100 as in the general experiment, and we vary the number of clusters from 5 to 25 with a step size of 5. We report results in Figure 2. As we can note, the number of clusters does not affect the general performance of the algorithm, so it is also very stable w.r.t. the setting of this parameter. We can observe that for *Airlines*, *Electricity*, *KDD99* and *Cover Type Sorted* datasets the maximum accuracy fluctuation over the different datasets is smaller than 0.8 points of accuracy. For the *Cover Type* dataset we reach the maximum accuracy gain (1.4 points of accuracy) between k=5 and k=25 when the budget is equal to 0.3. The maximum difference in accuracy (2.5 points of accuracy), among all the experiments, is measured for the *Poker* dataset for a budget of 0.15. For this dataset, *ACLStream* with a number of clusters equal to 5 always obtains the best accuracy. Still, even in this case, the performances of different numbers of clusters are very close to each other.

The obtained stability w.r.t. the number of clusters underlines that the combination of our macro and micro steps is effective for dealing with the complexity of data streams. Specifically, starting to analyze clusters with low homogeneity, which are clusters covering areas that are ambiguous according to prediction evidence, forces the method to sample examples where labeling information seems

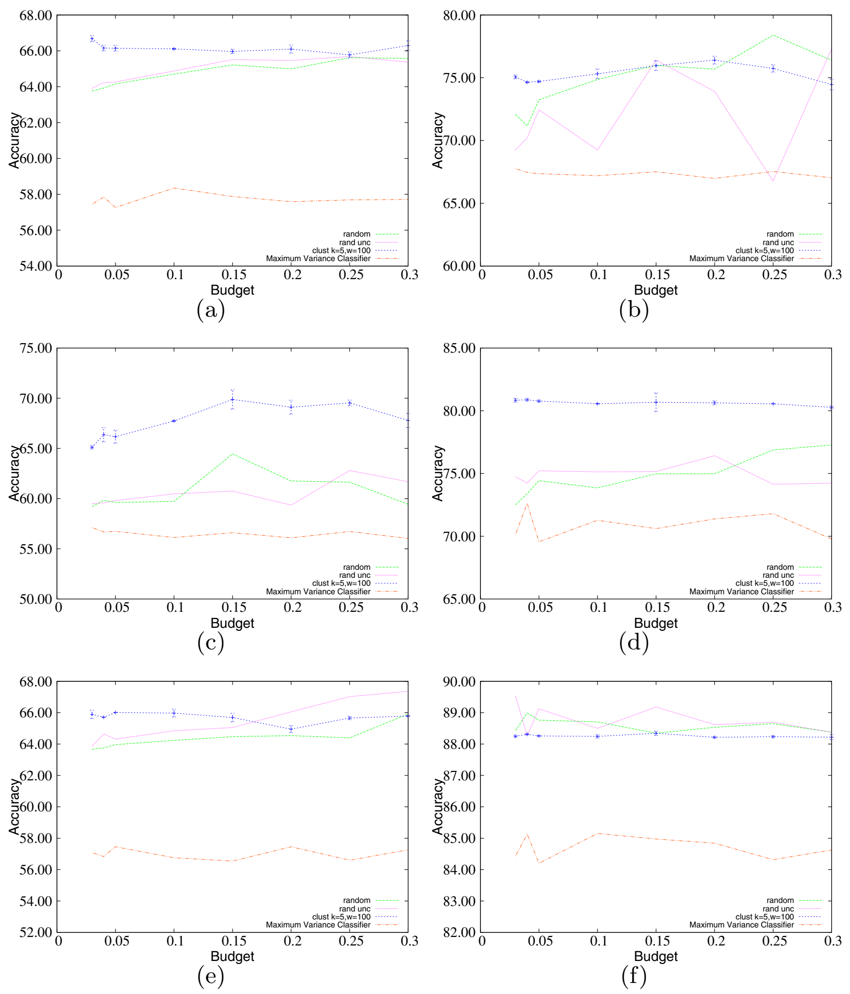


Fig. 1. Accuracy on a) Airlines b) Cover Type c) Poker d) Electricity e) Cover Type Sorted and f) KDD99

most useful. This is particularly important when the budget size is actually smaller than the total number of clusters.

4.4 Influence of the Batch Size

The last set of experiments focuses on the influence of the second parameter: the size of the batches. In particular, we analyze how the size of a batch impacts the final accuracy of *ACLStream*. For this purpose, we run experiments varying the size of the batches from 100 to 500 with a step size of 100. We average each result over 30 runs and we set the number of clusters equal to 5. The results are

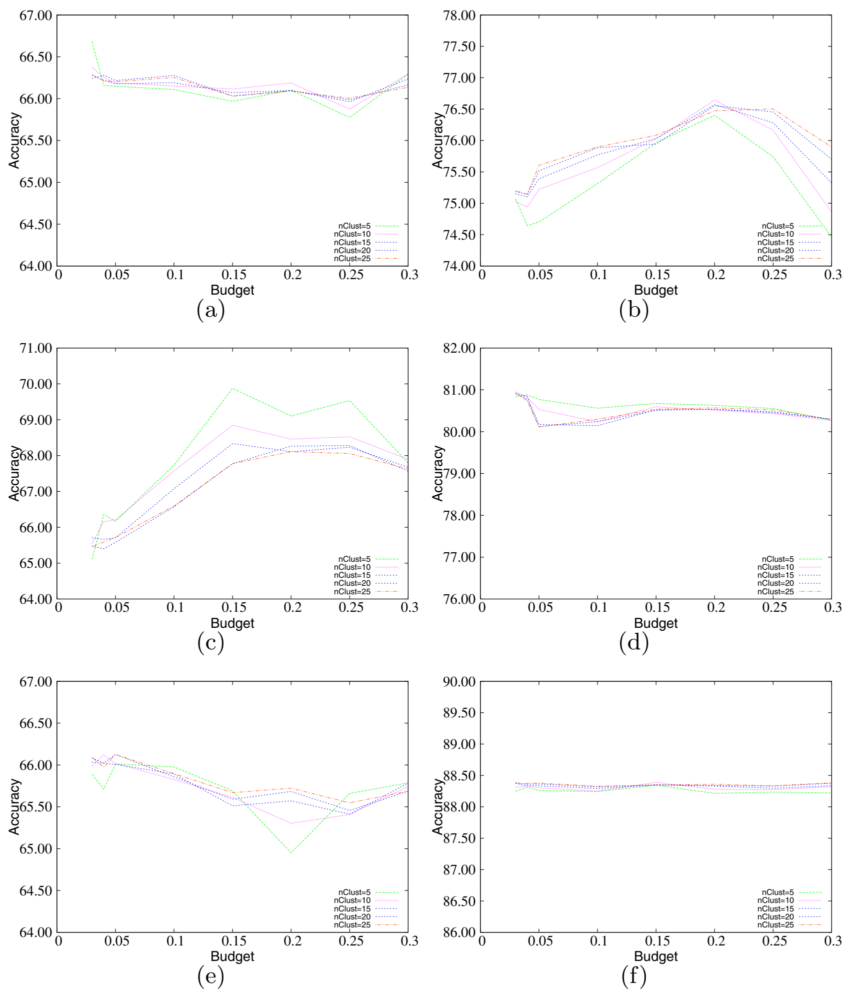


Fig. 2. Accuracy of *ACLStream* varying the number of clusters from 5 to 25 on a) *Airlines* b) *Cover Type* c) *Poker* d) *Electricity* e) *Cover Type Sorted* and f) *KDD99*

reported in Figure 3. We can observe two distinct behaviors. There is one group of three datasets – *Cover Type*, *Poker* and *Electricity* – for which the batch size influences the final accuracy, and another group comprising the remaining three datasets – *Airlines*, *Cover Type Sorted* and *KDD99*) – for which this parameter does not seem to affect the final performance. For the latter group of datasets the fluctuation in accuracy is less than 0.5 points. For the former group of datasets, the ones influenced by the batch size, we can note that in general smaller batches outperform bigger ones. Using small batches forces the learner to adapt faster to possible changes, and to sample instances in a more regular way. This is particularly useful where concept drift happens quickly and

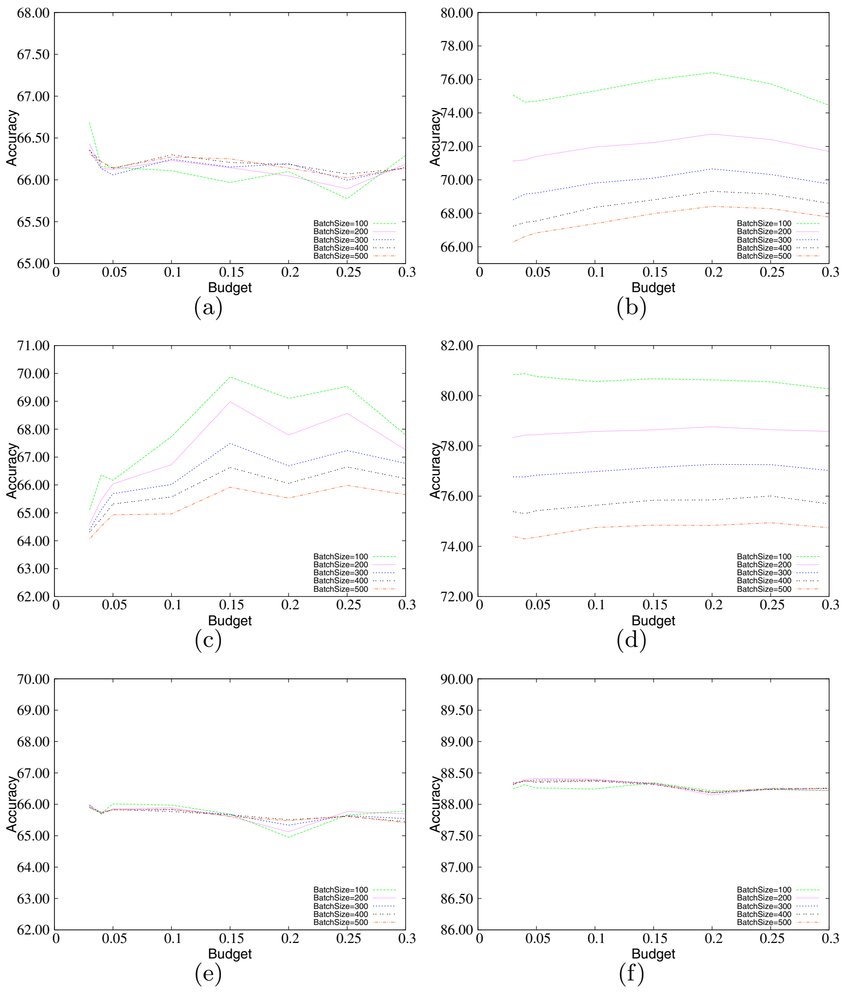


Fig. 3. Accuracy Results for *ACLStream* on a) *Airlines* b) *Cover Type* c) *Poker* d) *Electricity* e) *Cover Type Sorted* and f) *KDD99* varying the batch size from 100 to 500

frequently. This is the case for the *Electricty* dataset, where multiple levels of periodicity are present, over 24 hours, over 7 days, and over 4 seasons. Drift can be visually presented by plotting class conditional distributions over time. On the other hand, this is not the case for the *Cover Type Sorted* dataset, where batch size does not produce significant accuracy changes, as very gradual and smooth concept drift as induced by sorting the instances by the elevation attribute. This gradual drift phenomena can be managed well by all batch sizes evaluated here.

To summarize, for *ACLStream* we can state that small values for batch size are preferable over large ones. Using a small batch size forces the system to sample labels at more regular intervals. Consequently, the learner adapts better and faster in case of fast concept drift, without negatively impacting the performance on datasets where concept drift may be more gradual or not present at all.

5 Conclusions

Building classification models on data streams considering only a limited amount of labeled data is starting to be a common task due to time and cost constraints. In this paper we presented *ACLStream*, a novel algorithm to perform active learning in a data stream scenario. Our approach exploits a clustering based partitioning of the data space to focus sampling on the potentially most useful examples to label. Clusters are ranked by homogeneity of their predicted class distributions. Instances in each cluster are ranked according to two factors: i) maximum a posteriori classification probability, and ii) geometrical position inside the cluster. Certainty is defined to be the product of these two factors. Instances with low certainty inside a given cluster are preferred, as they represent central points over which the classifier is more uncertain. We assessed the performance of *ACLStream* over real world datasets and we showed that it outperforms state-of-the-art active learning strategies for data streams. We also empirically studied how our proposal is influenced by the setting of its parameters. As future work we would like to investigate in more detail the use of clustering for active learning in data streams considering alternative clustering algorithms as well as alternative ranking heuristics.

Acknowledgements. I. Žliobaitės research is supported by the Academy of Finland grant 118653 (ALGODAN).

References

1. Asuncion, A., Newman, D.: UCI machine learning repository, University of California, Irvine, School of Information and Computer Sciences (2007)
2. Attenberg, J., Provost, F.: Online active inference and learning. In: Proc. of the 17th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, KDD, pp. 186–194 (2011)
3. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive Online Analysis. J. Mach. Learn. Res. 11, 1601–1604 (2010)
4. Cesa-Bianchi, N., Gentile, C., Zaniboni, L.: Worst-case analysis of selective sampling for linear classification. J. Mach. Learn. Res. 7, 1205–1230 (2006)
5. Cohn, D., Atlas, l., Ladner, R.: Improving generalization with active learning. Machine Learning 15, 201–221 (1994)
6. Fan, W., Huang, Y., Wang, H., Yu, P.: Active mining of data streams. In: Proc. of the 4th SIAM Int. Conf. on Data Mining, SDM, pp. 457–461 (2004)
7. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 286–295. Springer, Heidelberg (2004)
8. Harries, M., Sammut, C., Horn, K.: Extracting hidden context. Machine Learning 32(2), 101–126 (1998)

9. Helmbold, D., Panizza, S.: Some label efficient learning results. In: Proc. of the 10th An. Conf. on Computational Learning Theory, COLT, pp. 218–230 (1997)
10. Huang, S., Dong, Y.: An active learning system for mining time-changing data streams. Intelligent Data Analysis 11, 401–419 (2007)
11. Ikonomovska, E., Gama, J., Dzeroski, S.: Learning model trees from evolving data streams. Data Mining and Knowledge Discovery 23(1), 128–168 (2010)
12. Klinkenberg, R.: Using labeled and unlabeled data to learn drifting concepts. In: IJCAI Workshop on Learning from Temporal and Spatial Data, pp. 16–24 (2001)
13. Lewis, D., Gale, W.: A sequential algorithm for training text classifiers. In: Proc. of the 17th An. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, SIGIR, pp. 3–12 (1994)
14. Lindstrom, P., Delany, S.J., MacNamee, B.: Handling concept drift in a text data stream constrained by high labelling cost. In: Proc. of the 23rd Int. Florida Artificial Intelligence Research Society Conference, FLAIRS (2010)
15. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Byers, A.: Big data: The next frontier for innovation, competition, and productivity (2011)
16. Masud, M.M., Gao, J., Khan, L., Han, J., Thuraisingham, B.: Classification and novel class detection in data streams with active mining. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010, Part III. LNCS, vol. 6119, pp. 311–324. Springer, Heidelberg (2010)
17. Masud, M., Woolam, C., Gao, J., Khan, L., Han, J., Hamlen, K., Oza, N.: Facing the reality of data stream classification: coping with scarcity of labeled data. Knowl. Inf. Syst. 33(1), 213–244 (2011)
18. Nguyen, H., Smeulders, A.: Active learning using pre-clustering. In: Proc. of the 21st Int. Conf. on Machine Learning, ICML, pp. 623–630 (2004)
19. Quinlan, R.J.: C4.5: Programs for Machine Learning. Kaufmann Series in Machine Learning. Morgan Kaufmann. Morgan Kaufmann (1993)
20. Sculley, D.: Online active learning methods for fast label-efficient spam filtering. In: Proc. of the 4th Conf. on Email and Anti-Spam, CEAS (2007)
21. Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley Longman Publishing (2005)
22. Widyantoro, D., Yen, J.: Relevant data expansion for learning concept drift from sparsely labeled data. IEEE Tr. on Know. and Data Eng. 17, 401–412 (2005)
23. Zhu, X., Zhang, P., Lin, X., Shi, Y.: Active learning from data streams. In: Proc. of the 7th IEEE Int. Conf. on Data Mining, ICDM, pp. 757–762 (2007)
24. Zliobaite, I., Bifet, A., Pfahringer, B., Holmes, G.: Active learning with drifting streaming data. IEEE Trans. on Neural Networks and Learning Systems (page in press, 2013)

Robust Crowd Labeling Using Little Expertise

Faiza Khan Khattak and Ansaf Salleb-Aouissi

Center for Computational Learning Systems (CCLS)
Columbia University, New York
fk2224@columbia.edu,
ansaf@ccls.columbia.edu

Abstract. *Crowd-labeling* emerged from the need to label large-scale and complex data, a tedious, expensive, and time-consuming task. But the problem of obtaining good quality labels from a crowd and their integration is still unresolved. To address this challenge, we propose a new framework that automatically combines and boosts bulk crowd labels supported by limited number of "ground truth" labels from experts. The ground truth labels help to estimate the individual expertise of crowd labelers and difficulty of each instance, both of which are used to aggregate the labels. We show through extensive experiments that unlike other state-of-the-art approaches, our method is robust even in the presence of a large proportion of bad labelers in the crowd. We derive a lower bound on the number of expert labels needed to judge crowd and dataset as well as to get better quality labels.

Keywords: Crowd-sourcing, Multiple labels, Ground truth.

1 Introduction

Crowd-sourcing is the gathering and leveraging of collective human intelligence to tackle tasks that cannot be readily automated. One example of a successful crowd-sourcing system is the reCAPTCHA project [22] for digitizing old books. This project leverages the power of human volunteers to transcribe approximately 500 million words at close to 100% accuracy, words that were otherwise unrecognizable by Optical Character Recognition (OCR) software. Another example of a widely used crowd-sourcing system is Amazon's Mechanical Turk (MTurk), which engages nearly thousands of workers registered to apply their brainpower to complex tasks including annotating medical images that contain malignant cells and identifying the videos suitable for a general audience.

In a crowd-labeling scene, an object is usually annotated by more than one person and the multiple labels are combined to produce one final label. While significant progress has been made on this process of aggregating crowd-labeling results, (e.g. [9],[19], [24]) it is well-known that the precision and accuracy of labeling can vary due to differing skill sets or even malicious behaviors of a labeler. Labelers can disagree about a specific label, requiring some type of mediation to resolve conflicting opinions. It is difficult to control for or determine in advance the proportion of low-quality/malicious labelers. If that proportion grows too high, there is often a phase transition leading to a steep,

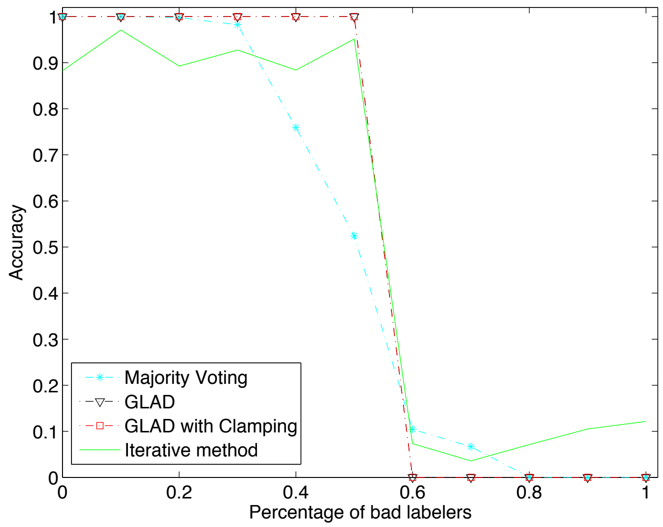


Fig. 1. Phase transition in the performance of majority voting, GLAD and GLAD with clamping [24] and iterative method [9] on UCI Mushroom dataset

non-linear drop in labeling accuracy as noted by Karger et al. [9]. It is also confirmed by our experiments on five different UCI datasets as we observe a phase transition in the performance of majority voting, GLAD and GLAD with clamping [24] and iterative method [9]. Figure 1 shows the phase transition for the UCI Mushroom dataset. This highlights the larger challenge of producing an objective assessment to measure the quality of the crowd for a given task.

One solution is to differentiate "Good Labelers" from the novice, careless or malicious labelers weighing their assessments more heavily in the final aggregation. Another important aspect that must be taken into consideration is the variability of the instance difficulty, which has received less attention in the crowd-sourcing literature.

We propose a new approach named Expert Label Injected Crowd Estimation (ELICE) which incorporates both metrics, expertise of crowd labeler and difficulty of the instance. ELICE uses only a few expert labels other than the crowd labels, to assess these metrics, which help to stabilize labeling and delay the phase transition to inaccurate labels. By experts, we mean the domain experts who provide ground truth. Acquiring expert labels is in most cases very expensive if not infeasible. However, acquiring crowd labels is cheap and easy, therefore, we get crowd labels for all instances.

Our main hypothesis is that using few expert-labeled instances, when available, tremendously benefits the crowd-labeling process. To assess the validity of our hypothesis, we have conducted an empirical evaluation that demonstrates the superiority of ELICE as compared to other state-of-the-art methods even when one injects only "few" expert labels in the labeling process. Our methodology appears to be robust even in the presence of a large proportion of low-quality labelers in the crowd. Furthermore, we derive a lower bound for the number of expert labels needed. This lower bound is a function of the overall quality of the crowd and difficulty of the dataset.

2 Related Work

Many recent works have addressed the topic of *learning from crowd* [18]. The most common and straightforward approach to aggregate crowd labels is majority voting. But the drawback of this method is that equal weights are assigned to the labels of each crowd labeler irrespective of his expertise. To overcome this problem of assigning equal weights to all the workers, different solutions are proposed. Many researchers use optimization methods such as Expectation-Maximization (EM) to solve this problem. In this context, Dawid & Skene [3] are the first to use EM algorithm for finding better quality labels as well as approximating the expertise of the labeler.

A probabilistic model of the labeling process is proposed by Whitehill et al. [24] named Generative model of Labels, Abilities, and Difficulties (GLAD). In this model EM is used to obtain maximum likelihood estimates of the unobserved variables and is shown to outperform majority voting. The authors also propose a variation of GLAD that *clamps* some known labels into the EM algorithm. More precisely, clamping is achieved by choosing the prior probability of the true labels very high for one class and very low for the other.

A probabilistic framework is also proposed by Yan et al. [25] as an approach to model annotator expertise and build classification models in a multiple label setting. A Bayesian framework is proposed by Raykar et al. [17] to estimate the ground truth and learn a classifier.

Sheng et al. [19], Sorokin et al. [21], and Snow et al. [20] show that using multiple, noisy labelers is as good as using fewer expert labelers. Active learning is used by Donmez et al. [5] to increase labeling accuracy choosing the most informative labels. This is done by constructing a confidence interval called "Interval Estimate Threshold" for the reliability of labeler. Also, Yan et al. [26] develop a probabilistic method based on the idea of active learning, to use the best labels from the crowd. An iterative approach is proposed by Karger et al. [9,10] which relies on a belief propagation algorithm to estimate the final labels weighted by each worker reliability. The authors use an explicit approach of instance assignment to labelers using a bipartite graph generated by random graph generation algorithm.

To handle adversarial labelers, Dekel & Shamir [4] propose the methodology that handles noisy labels and outperforms SVM classification, especially when the noise in the labels is moderate to high. In a crowd-labeling setting, identifying adversarial *labelers* is tackled through an a priori identification of those labelers before the labeling task starts, e.g. Paolacci et al. [15] but a malicious labeler can perform well on the test and then be adversarial again during the labeling process.

Similarly, the idea of using ground truth labels has been used by Crowdflower [13] who test the crowd expertise based on few ground truth. Their approach tests the crowd labelers during the training phase (before the actual labeling starts) and blocks the labelers who do not pass the training. Subsequent tests are also used to block bad crowd labelers after giving warnings. This is done by injecting instances for which ground truth is available during the actual labeling task. This approach can be helpful when a large number of ground truth instances which may not readily be available. To handle this problem Oleson et al. [14] propose "Progsrammatic gold" that generates gold units automatically which may not be possible for many datasets. Khattak and Salleb-Aouissi

[11,12] use ground truth to judge the crowd labeler expertise, instance difficulty as well as for inference of the final labels. The crowd labeler expertise are evaluated by adding one point for the correctly labeled instances and deducting one point for the mislabeled ones. Similarly the instance difficulty is estimated by adding one point for acquiring correct label from any labeler for an instance. These measure are then used to aggregate the crowd labels to get one final label.

The model proposed by Iperiotis et al. [7], [6] identifies the biased or adversarial labelers and corrects their assigned labels. This is done by replacing the *hard label* by a *soft label*. Class priors and the probability of a labeler assigning an instance from a particular class to some other class is used to calculate the soft labels.

ELICE has the ability to handle adversarial and below average labelers, in an integrated way. Instead of identifying them separately, we propose to acquire few expert labels for instances that are representative of the data, and judge the labelers and instance difficulty *after* the labeling task is achieved. Since our judgment is based only on a subset of instances and involves uncertainty about the estimates, we use entropy which allows to incorporate the uncertainty in the estimation of expertise of the labeler and difficulty of the dataset. We use these measures to calculate the trust we have in a label provided by a labeler for a particular instance. Based on this trust, we combine the crowd labels into one final label. This method helps in getting good approximation even when good labels are not available either because of the difficulty of the task or because of inexperienced labelers. Moreover, ELICE is computationally less expensive and is simple to implement.

This paper is organized as follows: in Section 3, we present our general framework. We derive in Section 4 a lower bound on the number of expert-labeled instances needed for ELICE. Section 5 is devoted to a variant of multi-class ELICE. Experiments demonstrating the efficiency and accuracy of our methodology on benchmarks and real-life datasets are provided in Section 6. We conclude with a summary and future work.

3 ELICE Framework

In this section ELICE and its cluster-based variant are described.

3.1 ELICE

Let $\mathcal{D}$ be a dataset of N unlabeled instances. We assign M crowd labelers to label the whole dataset; each instance i will receive a label $L_{ij} \in \{\pm 1\}$ from labeler j, where $i \in \{1, 2, \ldots, N\}$ and $j \in \{1, 2, \ldots, M\}$. To analyze the performance of the labelers, we get "ground truth" labels for a random sample $\mathcal{D}'(\subset \mathcal{D})$ of cardinality $n << N$. Instances of $\mathcal{D}'$ are labeled by one or more experts, which are assumed to be very knowledgeable and do not make mistakes. Therefore, only one expert label is acquired for each instance in $\mathcal{D}'$.

a) **Expertise of the labeler**
We use expert-labeled instances to evaluate the labelers by finding the probability of getting correct labels. This estimation of labeler's performance has a factor of

uncertainty since it is based on a sample. Therefore, entropy can be a natural way to measure this uncertainty. Entropy is high when the probability is around 0.5 as we are least certain about such a labeler and it is low when the probability is close to 0 or 1. The formula for the entropy for a worker j is given by:

$$E_j = -p_j log(p_j) - q_j log(q_j)$$

$$\text{such that}, \quad p_j = \frac{n_j^+}{n} \qquad q_j = 1 - p_j$$

$n_j^+ = |$correctly labeled instances from $\mathcal{D}'$ by labeler j$|$

Since we are more interested in the reliability of the assessment, we take $(1 - E_j)$. In order to differentiate between good and bad labelers, we multiply by $(p_j - q_j)$. This assigns a negative value to the bad labeler and positive value to the good one. We define the expertise of the labeler as

$$\alpha_j = (p_j - q_j)(1 - E_j) \tag{1}$$

where $\alpha_j \in (-1, 1)$

Multiplication by $(p_j - q_j)$ also allows for less variability in α_j when the number of correct and incorrect labels is close, assuming that it can be due to the choice of the instances in $\mathcal{D}'$. We can use α to categorize the labelers as follows:

- **Random guesser** is the labeler with α close to zero. This labeler is either a lazy labeler who randomly assigns the labels without paying any attention to the instances or an inexperienced labeler.
- **Good labeler** is the labeler with α close to 1. He does a good job of labeling.
- **Adversarial labeler** is the labeler with α close to -1. He guesses the correct label and then flips it.

b) **Difficulty of the instance**
Similarly, the difficulty of an instance is defined as:

$$\beta_i = (p'_i - q'_i)(1 - E'_i) + 1 \tag{2}$$

where $\quad p'_i = \frac{M_i^+}{M} \quad q'_i = 1 - p'_i$

p'_i is the probability of getting a correct label for instance i, from the crowd labeler, M_i^+ is the number of correct labels given to the instance i. Also,

$$E'_i = -p'_i log(p'_i) - q'_i log(q'_i)$$

represents the entropy for the instance i which measures the uncertainty in our assessment of the difficulty of the instance. All these values are calculated using the expert labeled instances.

We have added 1 to the formula in (2) because we find it more convenient mathematically to make the value of β positive. Another reason for adding 1 is that we cannot assume the difficulty level to be negative, just because the labelers did a bad job of labeling. We have $\beta_i \in (0, 2)$ which is used to categorize the instances as follows:

- **Easy instance** is the one with β close to 2.
- **Average difficulty instance** is the instance with β around 1.
- **Difficult instance** is the instance with β close to 0.

To judge the difficulty level of the remaining $(N-n)$ instances, we define *temporary labels* W_i as:

$$W_i = sign(\sum_{j=1}^{M} \alpha_j L_{ij})$$

The rest of β's are estimated by:

$$\beta_i = (p_i'' - q_i'')(1 - E_i'') + 1 \tag{3}$$

where p_i'', q_i'', E_i'' are calculated using the temporary labels.

c) **Aggregation of labels**
The parameters α, β are used to aggregate the labels. As a first step for this aggregation, we calculate the probability of getting a correct label for instance i from the labeler j defined as

$$P(T_i = L_{ij}|\alpha_j, \beta_i) = \frac{1}{(1 + exp(-3\alpha_j\beta_i))},$$

where T_i is the true but unknown label for the instance i. In this function, 3 is multiplied to span the values to the range (0,1), otherwise the values only map to a subinterval of (0,1).

The final approximation of the label is:

$$A_i = sign(\sum_{j=1}^{M} \frac{1}{(1 + exp(-3\alpha_j\beta_i))} L_{ij})$$

3.2 ELICE with Clustering

We propose a variation of ELICE called ELICE with clustering. Instead of picking the instances randomly from the whole dataset $\mathcal{D}$ for which we acquire expert labels, clusters of instances in $\mathcal{D}$ are first formed by applying k-means clustering using the features (if available), then equal number of instances are chosen from each cluster and given to the expert to label. This allows us to have expert labeled instances from different groups in the data, particularly when the dataset is highly skewed. Another possibility is to use any other method of clustering for instance K-means++ [2].

4 Bound on the Number of Expert Labels

Expert-labeled instances are used to learn labeler expertise α and difficulty of the instance β. Given that acquiring expert labels can be expensive and scarce, it is desirable

Table 1. Error distribution of the conjecture about the crowd and dataset. Crowd is categorized as good, average, and bad. Dataset is categorized as easy, mediocre, and difficult. Error can be high, medium, and low.

			Dataset		
			Very Difficult	**Moderate**	**Very Easy**
Crowd	**Very Bad**	Conjecture about Crowd	Bad	Bad–Avg.	Avg.–Good
		Conjecture about Dataset	Diff.	Diff.–Mod.	Diff.– Mod.– Easy
		Error	Low	Medium	High
	Average	Conjecture about Crowd	Bad–Avg.	Bad–Avg.–Good	Good– Avg.
		Conjecture about Dataset	Diff.–Mod.	Diff.–Mod.–Easy	Mod.–Easy
		Error	Medium	Medium	Medium
	Very Good	Conjecture about Crowd	Bad– Avg.	Good–Avg.	Good
		Conjecture about Dataset	Diff.–Mod	Diff.–Mod.	Easy–Mod.
		Error	High	Medium	Low

to find the lower bound of the expert labeled instances needed, which can also provide a good estimate of α and β. This scenario is similar to Probably Approximately Correct (PAC) learning where the learner has to learn the concept with a minimum number of examples with a given accuracy and confidence. Therefore, our approach to derive the bound is inspired by the PAC learning framework. As a prerequisite to this we estimate the distribution of error in our judgement about the crowd and instances.

The error distribution depends on the overall quality of the crowd and overall difficulty of the dataset, defined as follows:

- **Quality of the Crowd** (c) **:** Let p_j be the probability of getting a correct label from labeler j and f be the probability distribution of p_j. Then we define the quality of the crowd c as $c = E(P) = \sum_{j=1}^{M} p_j \; f(p_j)$
  Large values of c represent better crowd.

- **Difficulty of the Dataset** ($1 - d$) **:** We define, $d = E(Q) = \sum_{i=1}^{N} q_i \; h(q_i)$ where q_i is the probability of getting the correct label for instance i and h is the probability distribution for q_i. Higher d represents easier dataset.

In general, c and d are unknown, we make a conjecture about the crowd quality and dataset difficulty based on the performance of crowd on a given dataset. So the error depends on how much the conjecture deviates from the true values of c and d. Error is categorized as follows.

1. **High**: When the crowd is good and we conjecture it as a bad crowd (or vice versa) the error is high. This is also true when a dataset is easy and the conjecture is difficult (or vice versa).
2. **Medium**: When crowd is mediocre and we conjecture it as bad or good (or vice versa) then error is considered to be medium. Same is true about the dataset.
3. **Low**: Error is low when our judgment about the crowd and/or dataset is close to the true quality.

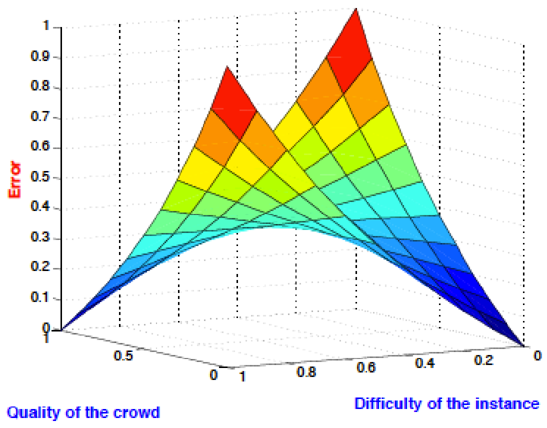


Fig. 2. Graph of the normalized error distribution. Quality of the crowd and difficulty of the dataset versus error.

The intuitive explanation of the error is summarized in Table 1 and described as follows:

a) **Good crowd & difficult instances:** When crowd is good and instances are difficult the performance of the crowd may be average. The conjecture made is that the crowd is bad to average and/or the dataset is of medium to high difficulty. So the error is high in this case.
b) **Bad crowd & difficult instances:** If crowd is very bad and instances are very difficult, then the performance of the crowd will be poor. Hence the conjecture will be bad crowd and/or difficult dataset. Therefore, the error is low.
c) **Good crowd & easy instances:** When crowd is very good and the instances are very easy our conjecture is good crowd and/or easy instances. Therefore, the error is low.
d) **Bad crowd & easy instances:** When crowd is bad and dataset is vey easy then the judgement can be biased and the error can be high.
e) **Average crowd OR Moderate instances:** When the crowd is of average capability then for any kind of the instances the judgment may not be very far from the true value hence the error is medium. This also holds for average difficulty dataset and any kind of crowd.

We formalize the relationship between the crowd, the dataset quality, and the error by the function:

$$e = \frac{1}{1 + (c - 1/2)(d - 1/2)}$$

When the crowd is below average i.e. c is less than $1/2$, we have $(c - 1/2) < 0$. When crowd is above average, we have $(c - 1/2) > 0$. This is also true for d. When the values of c and d are close to 1/2, $(c - 1/2)(d - 1/2)$ becomes small and hence e becomes high. When the values of c and d are close to 0 or 1 $(c - 1/2)(d - 1/2)$ is relatively larger so e is small. When one of the c or d is less than $1/2$ and the other is greater than $1/2$ then the value of e is average. The graph of the function is shown in Figure 2.

Theorem: For a given confidence $(1-\delta)$ and given values of c and d, the lower bound on the number of expert labels is given by

$$n \geq \frac{(b-a)(1+(c-1/2)(d-1/2))}{[1-a(1+(c-1/2)(d-1/2))]} log\ \frac{1}{\delta}$$

where a and b are the minimum and maximum of the values of the error e respectively.

Proof: The proof of this theorem is straight forward. We know that the number of examples required by a PAC learning model are given by

$$n \geq \frac{1}{\epsilon} log\ \frac{1}{\delta}$$

where ϵ is the error and δ is the level of confidence. In our case the error is depending on c and d hence the error e is here

$$e = \frac{1}{1+(c-1/2)(d-1/2)}$$

We normalize this error as follows

$$\epsilon = \frac{(e-a)}{(b-a)}$$

where a = min(e) & b = max(e) for $0 < c < 1$ and $0 < d < 1$.

Therefore, we get

$$\epsilon = \frac{1}{(b-a)}[\frac{1}{1+(c-1/2)(d-1/2)} - a]$$

Plugging the values in the PAC learning model we obtain the expression for the bound

$$n \geq \frac{(b-a)(1+(c-1/2)(d-1/2))}{[1-a(1+(c-1/2)(d-1/2))]} log\ \frac{1}{\delta}$$

□

5 ELICE Multi-class

ELICE can be easily extended to a generalized framework to handle multi-class classification, where we have $l > 2$ possible classes. Let L_{ij} be the label given to the ith instance by the jth labeler such that $L_{ij} \in \{1,2,3\ldots l\}$. As described in the previous section, we assume to have a dataset of N instances labeled by M crowd labelers. We have $n << N$ expert labeled instances. We proceed by finding α_j's and β_i's using the formulas in equations (1) and (2). After finding α_j's for all the M crowd labelers and

β_i for n expert-labeled instances, we use these to infer *temporary labels* W_i for the rest of $(N - n)$ instances which are not labeled by the expert.

$$W_i = \text{index of } \max\{S_1, S_2, \ldots, S_l\}$$

$$\text{where} \quad S_k = \begin{cases} S_k + \alpha_j & \text{if } L_{ij} = k \\ S_k & \text{if } L_{ij} \neq k \end{cases} \quad \text{where} \quad j = 1, \ldots, M, \quad k = 1, \ldots, l$$

and S_k's are initialized with zero.
Temporary labels W_i are used to find β_i for the rest of the $(N - n)$ instances using equation (3). Final labels A_i are inferred using the following formulas.

$$A_i = \text{index of } \max\{\sigma_1, \sigma_2, \ldots, \sigma_l\}$$

such that,

$$\sigma_k = \begin{cases} \sigma_k + \frac{1}{(1+exp(-3\alpha_j\beta_i))} & \text{if } L_{ij} = k \\ \sigma_k & \text{if } L_{ij} \neq k \end{cases} \quad \text{where } j = 1, \ldots, M \quad k = 1, \ldots, l$$

Also, σ_k's are initialized with zero.

6 Experiments

We have implemented ELICE with its variants along with Majority Voting and the iterative method [9] in Matlab. We acquired the code for GLAD and GLAD with clamping released by Whitehill et al. [24]. In Section 6.1, we present results of our experiments on five datasets from the UCI repository [1]. We compare the accuracy and efficiency of ELICE to GLAD, GLAD with clamping, iterative approach, and majority voting. We also provide results for ELICE with clustering for the UCI datasets as they have features available, which are used for clustering. Crowd labels were simulated for different rates of bad crowd labelers. In Section 6.2 and Section 6.3, we compare the accuracy of our method to the other state-of-the-art approaches on two real applications for which we used MTurk to acquire labels from the crowd.

6.1 UCI Datasets

In this experiment, we selected five datasets from the UCI repository: Mushroom, Chess, Tic-Tac-toe, Breast cancer and IRIS (with this latter restricted to 2 classes). We simulated 20 crowd labels for each instance in these experiments. The labels were generated so that a good crowd labeler makes less than 20% mistakes. These were created by inverting $x\%$ of the original labels in the dataset, where x is a random number between 0 and 20. Bad crowd labelers were assumed to make more than 80% mistakes and their labels were generated in a similar way. We randomly selected n number of instances to play the role of the expert-labeled instances.

We ran ELICE and its variant with clustering, along with the other methods. Table 1 shows a comparison of accuracy of different methods as compared to ELICE across

Table 2. Accuracy of Majority voting, GLAD (with and without clamping) and ELICE (with and without clustering) for different datasets and different rates of bad labelers. Given results are the average of 100 runs.

Bad Labelers	Dataset($\mathcal{D}$)	Mushroom	Chess	Tic-Tac-Toe	Breast Cancer	IRIS
	Total instances (N)	8124	3196	958	569	100
	+ve/-ve instances	3916/ 4208	1669/1527	626/332	357/212	50/50
	Expert labels(n)	20	8	8	8	4
Less than 30%	Majority Voting	0.9963	0.9877	0.994	0.9969	0.9925
	GLAD	1.0000	1.0000	1.0000	1.0000	1.0000
	GLAD with clamping	1.0000	1.0000	1.0000	1.0000	1.0000
	Iterative algorithm	0.9118	0.9300	0.9619	0.9073	0.9300
	ELICE	1.0000	0.9996	0.9999	1.0000	1.0000
	ELICE with clustering	0.9999	0.9995	0.9999	1.0000	1.0000
30 to 70%	Majority Voting	0.4121	0.3166	0.3907	0.402	0.36
	GLAD	0.5001	0.2502	0.5003	0.5004	0.2552
	GLAD with clamping	0.5002	0.2502	0.2508	0.5004	0.2552
	Iterative algorithm	0.5473	0.2596	0.2873	0.4982	0.2925
	ELICE	0.9751	00.9727	0.9574	0.9372	0.986
	ELICE with clustering	0.9656	0.9704	0.9526	0.9356	0.9852
More than 70%	Majority Voting	2.46E-04	0.00041	0.001	0.0029	0.01
	GLAD	1.23E-04	3.14E-04	0.001	0.0018	0.0104
	GLAD with clamping	1.23E-04	3.14E-04	0.001	0.0018	0.0104
	Iterative algorithm	0.0681	0.0835	0.1089	0.0463	0.0567
	ELICE	0.236	0.1957	0.1952	0.1939	0.2506
	ELICE with clustering	0.2005	0.1508	0.1731	0.1699	0.2577

the five datasets for different rates of bad labelers. Note that our methods outperform majority voting, GLAD, GLAD with clamping [24] and Iterative method [9] even when the percentage of bad labelers is increased to more than 70%. The graph showing the accuracy for the Mushroom dataset is presented in Figure 3. The experiments also revealed that ELICE is efficient as compared to the other methods. Figure 4 shows the runtime for Mushroom for all the methods as we increase the number of instances.

6.2 Race Dataset

To test our approach on a real-life dataset, we considered a *race recognition* dataset[1]. We took three samples of 100 instances each and posted them as race recognition tasks on Amazon Mechanical Turk. The samples were chosen to guarantee different levels of difficulty. The tasks were to identify: (1) Black versus Caucasian (50 instances of each class), (2) Hispanic versus Asian (50 instances of each class), (3) Multiracial versus other races (40 instances of Multiracial and 60 instances of the other races i.e. Asian, Black, Caucasian and Hispanic.) Snapshots of the experiment as posted on MTurk are shown in Figure 5.

[1] Available on Stimulus Images; Courtesy of Michael J. Tarr, Center for the Neural Basis of Cognition, Carnegie Mellon University `http://tarrlab.cnbc.cmu.edu/face-place`.

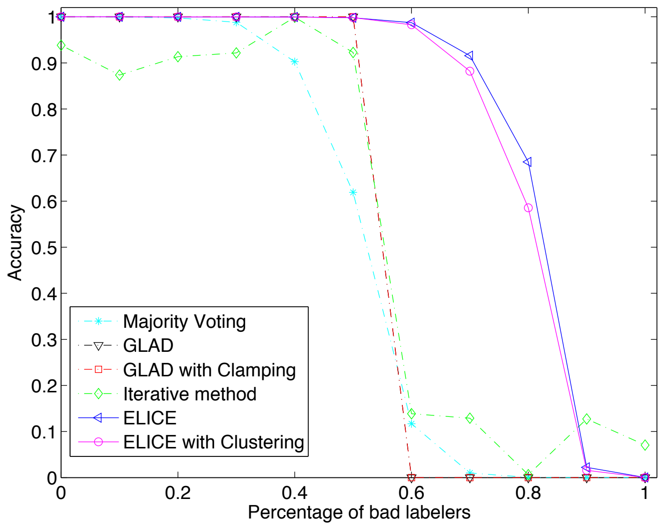


Fig. 3. Accuracy vs. percentage of bad labelers. Number of expert labels used for ELICE and ELICE is 20.

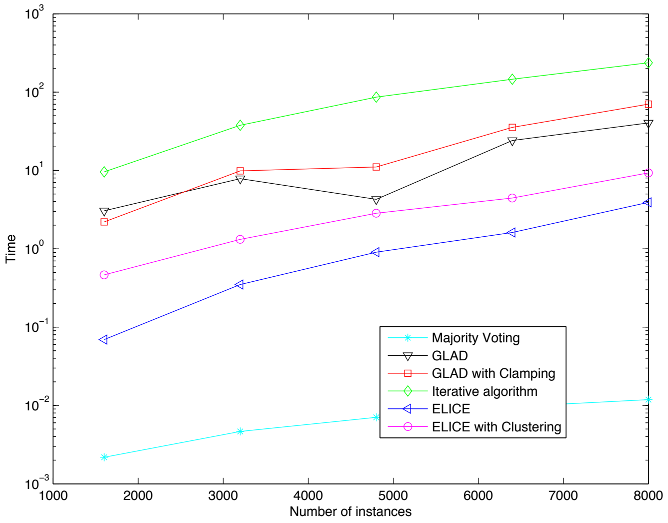


Fig. 4. Time vs. Number of instances. Number of expert labels used for ELICE and ELICE is 20.

For each task, we acquired six crowd labels for all 100 instances. The three tasks were chosen to guarantee easy to moderate difficulty level. Black versus Caucasian was the easiest of the tasks. Therefore, most of the labelers performed really well with only 0% to 25% of mistakes. Identifying Hispanic versus Asian was relatively more difficult. In this case, good labelers made less than 15% mistakes and bad labelers made

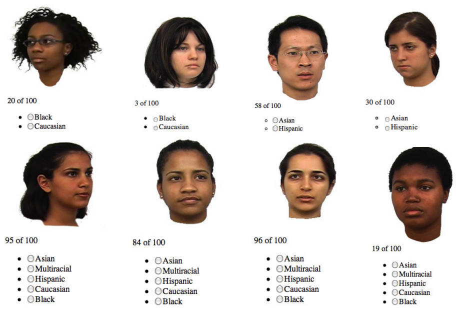


Fig. 5. Example images from the Race recognition task posted on Amazon Mechanical Turk (Left to right): (Top) Black, Caucasian, Asian, Hispanic. (Bottom) Multiracial, Hispanic, Asian, Multiracial.

Table 3. Accuracy of different methods on Amazon Mechanical Turk datasets. Given results are the average of 10 runs on 100 instances with 6 labels per instance. Randomly chosen 8 instances are used as expert labeled instances (the instances with ground truth.)

	Race			**Breast Cancer**
	Black/Caucasian	Hispanic/Asian	Multiracial/other	Malignant/Non-malignant
MajorityVoting	0.95	0.52	0.650	0.62
GLAD	0.8587	0.5	0.663	0.3152
GLAD with Clamping	0.8587	0.5	0.663	0.3043
Iterative	0.5	0.47	0.5573	0.3982
ELICE	**0.9562**	**0.6225**	**0.667**	**0.6858**

over 48% mistakes. But the most challenging of all was identifying multiracial from the other races. While most of the labelers did equally bad, surprisingly it was not as bad as we had expected: the percentage of mistakes ranged between 30% and 50%. Here also, we ran experiments using the state-of-the-art methodologies along with our methods. For all variants of ELICE, we used 8 random instances for which we had ground truth. The results are shown in Table 3. Here again, the experiment revealed that ELICE outperforms the rest of the methodologies in estimating the right labels.

6.3 Breast Cancer Dataset

Early identification of breast cancer can help in preventing thousands of deaths but identifying cancer is not an easy task for untrained eyes. We posted 100 mammograms [2] on Amazon Mechanical Turk. The task was to identify Malignant versus others (Normal, Benign, Benign without call back.). The following instruction for appropriate identification was provided to the labelers: “A breast tumor is a dense mass and will appear whiter than any tissue around it. Benign masses usually are round or oval in shape, but a tumor may be partially round, with a spiked or irregular outline as part of its circumference.”

[2] `http://marathon.csee.usf.edu/Mammography/Database.html`.

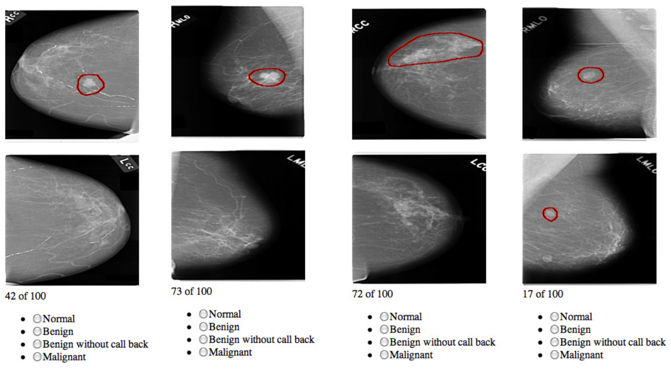


Fig. 6. Example images of the breast cancer dataset. From left to right: First three are Malignant and fourth is benign.

This task clearly requires expertise and is very difficult for an untrained person. On the other hand, an expert person can do really well but is very expensive. In this case, we had two labelers with less than 33% mistakes and four labelers with more than 55% mistakes. Results are shown in Table 3 and demonstrate once again the superiority of ELICE as compared to the other methods when injecting only 8 expert labels in the labeling process.

7 Conclusion and Future Work

We present a new method, named ELICE, demonstrating the benefit of using "few" expert labels in a multiple crowd labeling setting, which improves the estimation of the ground truth. ELICE is efficient and effective in achieving accuracy for *any* setting including when crowd quality is heterogeneous. We show through several experiments on real and synthetic datasets that unlike other state-of-the-art methods, our method is robust even in the presence of large number of bad labelers. One of the most important aspect of our method is overcoming the phase transition inherent in other approaches. We also derive a lower bound on the number of expert labels needed.

So far, we have been dividing the instances into equal-sized subsets and assign a fixed number of crowd labelers to label these subsets. An equal number of instances is then drawn from each subset to be labeled by the expert. One can use a more elaborated approach of instance assignment as done in [9,10]. A bipartite graph is used to model the instances and labelers. The graph is assumed to be regular but not complete so as not every instance gets labeled by every labeler. Then a subset can be chosen to get expert labels such that every crowd labeler is evaluated on equal number of instances. We propose a variant of ELICE using clustering as one way to achieve this. Another possibility is to ask the expert to identify and label difficult instances. Also, crowd can be used to point out the difficult instances to get expert labels [23]. Recently, Ho et al. [8] used primal dual method for task assignment and label inference. Future work will apply ELICE to highly imbalanced datasets and to a word-sense annotation task using multiple annotators [16].

References

1. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
2. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, pp. 1027–1035. Society for Industrial and Applied Mathematics, Philadelphia (2007)
3. Dawid, A.P., Skene, A.M.: Maximum likelihood estimation of observer error-rates using the em algorithm. Applied Statistics 28, 20–28 (1979)
4. Dekel, O., Shamir, O.: Good learners for evil teachers. In: ICML, p. 30 (2009)
5. Donmez, P., Carbonell, J.G., Schneider, J.: Efficiently learning the accuracy of labeling sources for selective sampling. In: KDD, pp. 259–268 (2009)
6. Ipeirotis, P.G., Paritosh, P.K.: Managing crowdsourced human computation: a tutorial. In: WWW (Companion Volume), pp. 287–288 (2011)
7. Ipeirotis, P.G., Provost, F., Wang, J.: Quality management on amazon mechanical turk. In: Proceedings of the ACM SIGKDD Workshop on Human Computation (2010)
8. Ju Ho, C., Jabbari, S., Vaughan, J.W.: Adaptive task assignment for crowdsourced classification. In: Proceedings of the 30th International Conference on Machine Learning (ICML 2013). JMLR Workshop and Conference Proceedings, pp. 534–542 (2013)
9. Karger, D., Oh, S., Shah, D.: Budget-optimal task allocation for reliable crowdsourcing systems. CoRR (2011)
10. Karger, D., Oh, S., Shah, D.: Iterative learning for reliable crowdsourcing systems. In: NIPS, Granada, Spain (2011)
11. Khattak, F.K., Salleb-Aouissi, A.: Quality control of crowd labeling through expert evaluation. In: Second Workshop on Computational Social Science and the Wisdom of Crowds, NIPS, Granada, Spain (2011)
12. Khattak, F.K., Salleb-Aouissi, A.: Improving Crowd Labeling through Expert Evaluation. In: AAAI Symposium on the Wisdom of the Crowd (2012)
13. Le, J., Edmonds, A., Hester, V., Biewald, L., Street, V., Francisco, S.: Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In: Evaluation, pp. 17–20 (2010)
14. Oleson, D., Sorokin, A., Laughlin, G.P., Hester, V., Le, J., Biewald, L.: Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. In: Human Computation. AAAI Workshops, WS-11-11. AAAI (2011)
15. Paolacci, G., Chandler, J., Ipeirotis, P.G.: Running experiments on amazon mechanical turk. Judgment and Decision Making 5(5), 411–419 (2010)
16. Passonneau, R.J., Salleb-Aouissi, A., Ide, N.: Making sense of word sense variation. In: Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions, DEW 2009, pp. 2–9. Association for Computational Linguistics, Stroudsburg (2009)
17. Raykar, V., Yu, S., Zhao, L., Jerebko, A., Florin, C., Valadez, G., Bogoni, L., Moy, L.: Supervised learning from multiple experts: whom to trust when everyone lies a bit. In: ICML 2009, pp. 889–896 (2009)
18. Raykar, V.C., Yu, S., Zhao, L., Valadez, G.H., Florin, C., Bogoni, L., Moy, L.: Learning from crowds. JMLR 11, 1297–1322 (2010)
19. Sheng, V., Provost, F., Ipeirotis, P.: Get another label? Improving data quality and data mining using multiple, noisy labelers. In: KDD 2008, pp. 614–622 (2008)
20. Snow, R., O'Connor, B., Jurafsky, D., Ng, A.: Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In: EMNLP 2008, pp. 254–263. Association for Computational Linguistics, Morristown (2008)

21. Sorokin, A., Forsyth, D.: Utility data annotation with amazon mechanical turk. In: Computer Vision and Pattern Recognition Workshops (January 2008)
22. von Ahn, L., Maurer, B., Mcmillen, C., Abraham, D., Blum, M.: recaptcha: Human-based character recognition via web security measures. Science 321(5895), 1465–1468 (2008)
23. Wallace, B.C., Small, K., Brodley, C.E., Trikalinos, T.A.: Who should label what? Instance allocation in multiple expert active learning. In: In Proc. of the SIAM International Conference on Data Mining, SDM (2011)
24. Whitehill, J., Ruvolo, P., Wu, T., Bergsma, J., Movellan, J.: Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In: NIPS, pp. 2035–2043 (2009)
25. Yan, Y., Rómer, R., Glenn, F., Mark, S., Gerardo, H., Luca, B., Linda, M., Jennifer, G.: Modeling annotator expertise: Learning when everybody knows a bit of something. In: AISTAT (2010)
26. Yan, Y., Rosales, R., Fung, G., Dy, J.: Active learning from crowds. In: ICML 2011, pp. 1161–1168. ACM (2011)

A New Approach to String Pattern Mining with Approximate Match

Tetsushi Matsui[1], Takeaki Uno[1], Juzoh Umemori[2], and Tsuyoshi Koide[2]

[1] National Institute of Informatics
2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
{tetsushi,uno}@nii.jp
[2] National Institute of Genetics, Mishima, Shizuoka 411-8540, Japan

Abstract. Frequent string mining is the problem of finding frequently appearing strings in a given string database or large text. It has many applications to string data analysis on strings such as texts, word sequences, and genome sequences. The problem becomes difficult if the string patterns are allowed to match approximately, i.e., a fixed number of errors leads to an explosion in the number of small solutions, and a fixed ratio of errors violates the monotonicity that disables hill climbing algorithms, and thus makes searching difficult. There would be also a difficulty on the modeling of the problem; simple mathematical definitions would result explosion of the solutions. To solve this difficulty, we go back to the motivations to find frequent strings, and propose a new method for generating string patterns that appear in the input string many times. In the method, we first compute the similarity between the strings in the database, and enumerate clusters generated by similarity. We then compute representative strings for each cluster, and the representatives are our frequent strings. Further, by taking majority votes, we extend the obtained representatives to obtain long frequent strings. The computational experiments we performed show the efficiency of both our model and algorithm; we were able to find many string patterns appearing many times in the data, and that were long but not particularly numerous. The computation time of our method is practically short, such as 20 minutes even for a genomic sequence of 100 millions of letters.

1 Introduction

String data is one of the most popular data types, including texts, word sequences, genome sequences, and so on. When we analyze string data, we sometimes want to find frequently appearing string patterns, called frequent strings, in the data. In the bioinformatics field, such frequent strings are called "motifs", and many studies have been done on the algorithms for finding motifs. Frequent string mining is the task to find frequent strings. The problem of finding patterns frequently appearing in a given database is called frequent pattern mining, and widely studied. The recent development of algorithms enables us to handle large scale databases, as shown in the implementation competitions[4].

More structured patterns such as sequences and graphs are also considered, and many algorithms are proposed [6,10,15].

Actually, there are at most $n(n+1)/2$ substrings for a string of n letters, thus frequent strings can be enumerated in polynomial time. Moreover, constructing a suffix array[7] makes it possible to obtain all the frequent strings in a compact form in $O(n)$ time. This computational efficiency has enhanced development in many areas in this field, such as natural language processing.

On the other hand, for application to real-world data, we have to allow approximate matches, i.e., matches that include errors. In natural languages, many ambiguities occur in writing, and numerous human errors such as typos also occur. As a basic principle, genome sequences include errors. Consequently, it is exceedingly unlikely that long subsequences coming from the same origin will be exactly identical. In such cases, exact matches can give only short and trivial patterns. Unfortunately, frequent pattern mining cannot well deal with approximate matching for finding large patterns. For example, if we allow patterns to have at most θ errors in matching, huge number of short strings become frequent, since any string of θ letters can match anything in the string data. This disturbs the usual hill climbing algorithms, since they in principle have to explore all small patterns.For example, the algorithm of Mitasiunaite and Boulicaut[8] took one hour to one day, to extract frequent strings of length 5 to 10 with 2 or 3 errors, from string data of millions of letters. It would be very hard to find string patterns of length more than 100 with up to 10 errors by this approach.

On the other hand, if we allow errors according to the pattern length, such as ρl mismatches for strings of length l with a ratio ρ, hill climbing algorithms do not work since patterns do not satisfy the anti-monotone property. Fundamentally, frequent pattern mining with approximate match (soft occurrence) has a difficulty on the definition of the problem, since even if we ignore small patterns, there might be huge number of large frequent patterns. For example, when string "AAAA...AAA" is included in many times in the data, any string with short distance from it, such as "BBA...AAA" and "BAZ...AA", can be a frequent string. We should formulate the problem so that only some representatives of such strings will be output, however, defining representatives is difficult. Pattern mining aims to enumerate patterns interesting/valuable to human being, and its good mathematical definition is hard.

In this paper, we propose a new "method" for generating string patterns that may appear in the given string data many times. That is, we define our task, but do not formulate the problem precisely. Instead of that, we propose a process to generate string patterns from the input string data. Actually, many algorithms in bioinformatics take this approach (see a survey[13]). This approach is often taken in areas with mathematically undefined objectives, such as information retrieval, and web science. In algorithmic area, heuristic algorithms can be considered as this approach. For example, greedy algorithms have well-defined objectives, but their output can not be formulated as a simple mathematical terms. The algorithms are description of the output.

To design our algorithm, we observe that if a string approximately matches some substrings in the string data, the substrings are relatively similar to each other. This observation leads us that clusters of substrings generated by similarity can be regarded as the set of substrings that some frequent patterns match. On this basis, we enumerate the clusters first and then compute their representatives. The representatives are our model of frequent strings, and we show its completeness. We can prove that for any frequent string defined in a usual way, there is a representative not so different from the pattern. Furthermore, by taking majority votes, we extend the frequent strings in both directions to obtain longer frequent strings.

There are actually similar approaches in bioinformatics[5,12,13] However, because of the computational difficulty, they can deal only with small data, can find few patterns, or have some heuristics such as removing very frequent small strings from the data, that loses completeness and disables to deal with other string data. Moreover, they do not use enumerational approaches, thus the obtained patterns usually do not have the completeness. For example, let us see HomologMiner[5] that is designed to find clusters of similar substrings. It first removes all frequently appearing short strings called *repeat sequences* from the input genome sequences, by looking at the library of repeat sequences. It then computes the similarity between all substrings, and cluster the substrings. Because of the heavy pairwise comparison (even though they avoid heavy DP), they needed up to ten hours. δ-free pattern mining aims to reduce the number of patterns by pruning patterns including patterns with similar occurrences[3], but the reduction ratio is limited, say at most 1/10 in noisy data.

To achieve a high computational performance, we keep our task to be simple. We address two tasks in this paper; find short string patterns that have Hamming distances of at most the given threshold d to many substrings of the input string data, and to find many string patterns that have short edit distances to many substrings of the input string data. The former can be seen as motifs, and the latter can be seen as consensus sequences, that have been extensively studied in bioinformatics. By using Hamming distance as error criterion, we can find similar substring pairs in a very short time. There are several algorithms for this task, especially in bioinformatics such as BLAST[1,2] and FASTA[9]. Among these, we chose the algorithm of [14] because of its high performance. Actually, the other algorithms basically use exact matches to find approximate matches, thus they do not fit our purpose; if we use these algorithms, we should use exact match directly, to make everything clear and simple. The algorithm we chose can terminate in a few minutes even for 10 million of genome sequences of 30 letters with Hamming distance threshold 2.

Our computational experiments show that our algorithm runs in a practically short time, say 10 minutes, even for large scale data of up to one million letters, for many kinds of real world data such as genome sequences, natural language texts, and word sequences. The patterns found by our method are long, and their number is quite small compared to those found by usual frequent pattern mining. The number of patterns found is usually in the order of 100 or 1,000.

This range is suitable for the practical use of pattern mining, i.e., ten is too small (it should be solved by optimization approaches), and one million is too much.

2 Preliminaries

Let Σ be an alphabet of letters, and a *string* be a sequence of letters. The *length* of a string S is the number of letters in S and is denoted by $|S|$. The string of length zero is also a valid string and is called the *empty string*. The ith letter of a string S is written $S[i]$, and i is called the *position* of $S[i]$. The substring of S starting from the ith letter and ending at the jth letter is a string $S[i]S[i+1]\ldots S[j]$, and is denoted by $S[i,j]$. When $i < 1$ or $i > |S|$, $S[i]$ is a special letter '\$' that is not in Σ. For example, when string S is $ABCDEFG$, $S[3] = C$, and $S[4,6] = DEF$. When $j < i$, we define $S[i,j]$ to be the empty string, and when $i < 1$, the $S[i,j]$ starts with '\$'. For two strings S_1 and S_2, the *concatenation* of S_2 to S_1 is a string S given by appending S_2 to the tail of S_1, i.e., $|S| = |S_1| + |S_2|$, $S[i] = S_1[i]$ if $i \leq |S_1|$, and $S_2[i - |S_1|]$ otherwise.

For two strings S_1 and S_2 of the same length, the *Hamming distance* of S_1 and S_2 is defined by the number of positions i satisfying $S_1[i] \neq S_2[i]$. Such letters are called the *mismatches* of S_1 and S_2, and the positions of mismatches are called the *mismatch positions* of S_1 and S_2. The *edit distance* (Levenshtein distance) between S_1 and S_2 is defined by the minimum number of operations to transform S_1 to S_2, where the operations are to change/insert/delete one letter. Suppose that we are given a distance measure. For a string S, the substrings in the string data that have a short distance to S are called the *occurrences* of S. A set of occurrences of S is called an *occurrence set* of S.

3 Model and Method

As we discussed in the introduction section, the formulation itself is a difficult problem. There may be a need for a paradigm shift in the problem formulation, one that would be mining process based. Since such a new paradigm should come from observing the real-world applications and the structures of frequent patterns, we begin with discussing the requirements for models of frequent strings.

(1: simplicity) *Distance for evaluating the similarity has to be simple.*
The definition of "distance" affects the performance of both models and algorithms. Here we suppose to keep it simple, for the computational efficiency and easy understanding of the models and results. For the use of more sophisticated distances, we can use the solutions obtained with a simple distance as candidates for the frequent strings, since we can expect if two strings are similar in a sophisticated distance, they are also similar in a simpler distance.

(2: independence) *The output solutions do not have huge patterns similar to each other.*

Pattern mining often outputs many similar solutions. This usually disturbs the analysis, i.e., it makes the output difficult to understand, and disables fast post processes.

(3: completeness) *The set of patterns should be complete.*
One motivation in carrying out pattern mining is the possibility to find previously undiscovered knowledge. In such cases, we do not want to miss any important pattern, but on the other hand we will be satisfied if any important pattern has some similar patterns in the output. Hence, some completeness of output solutions is needed, i.e., algorithms should not miss patterns similar to no output pattern.

(4: representativeness) *Patterns have to represent some aspects of data.*
Even if a string pattern is frequent, it sometimes is not valuable. For example, if we allow Hamming distance of two, string AA matches everywhere in the data, but it makes no sense. Particularly, we usually want to obtain rare patterns that are hidden in the usual mean-variance analysis, thus patterns have to match not so many substrings, that compose a local structure.

3Considering condition (1), we presume to use either the edit distance or the Hamming distance. Since this paper addresses large scale problems, we use the Hamming distance for purpose of the computational efficiency. Our computational experiments show that long string patterns can be obtained from short string patterns found with Hamming distance, and they approximately match many substrings in the sense of edit distance. HomologMiner also uses Hamming distance[5], and could find similar genome sequences efficiently.

As mentioned above, it frequently happens that simple mathematical formulations of frequent pattern mining either do not satisfy certain conditions, or are computationally inefficient. In particular, condition (2) is often violated[4]. When many exactly identical substrings exist in the data, all the strings that are only a short distance from the substring become frequent strings. This allows a small number of similar substrings groups can generate numerous patterns, thus invalidating the "finding all frequent patterns" approach. It is observed from that the number of frequent patterns drastically increases if approximate matches are allowed[8]. Heuristic search methods often violate condition (3) (for example [5]), since it takes much time to obtain one solution, and it is difficult not to miss any significant solution, by ad hoc repetitions of the search.

These conditions provide certain principles to which frequent strings should adhere. The first principle comes from the definition of "occurrence", and the condition (1).

(α) The occurrences of string S that matches many substrings are relatively similar to each other, since they all have short distances to S.
(β) Two different frequent substrings should not have the same, or quite similar, occurrence sets. Rather, an occurrence set should be induced by one pattern.

When the distance measure satisfies the triangle inequality, any two strings with distances at most d from S satisfy that the distance between them is at most $2d$. Principle (β) comes from conditions (2) and (4); many frequent strings should not be induced by the same set of occurrences, and existence of similar occurrence sets would induce unnecessarily many similar frequent strings.

These principles motivate us to approach the problem from the occurrence sets; the number of patterns is quite huge, but the possible combinations of occurrences are limited and thereby tractable. The principle (α) implies that a set of substrings similar to each other, which we call a *cluster*, can be a candidate of an occurrence set, and (β) implies that a string similar to every substring in the cluster would be a frequent string. This is a reverse course from the usual mining process, which first generates patterns and then computes their occurrences. In our approach the occurrence sets are computed first, and then patterns are induced from the occurrence sets. By enumerating clusters, we can obtain in some sense completeness of the frequent strings. This enables mining problems to be formulated as process-based problems.

Problem: Cluster-based Frequent String Mining
input string set $\mathcal{D} = \{S_1, \ldots, S_m\}$
1. (Cluster enumeration) Find groups (clusters) of substrings of strings in $\mathcal{D}$ such that the substrings in the group are similar to each other.
2. (Centroid computation) Compute a string similar to all the substrings in a cluster, for each cluster.

This would involve a computational difficulty on the cluster enumeration step if the length of strings to be found was long. To cope with this difficulty, we observe the following principle.

(γ) Long similar strings have a common short string that is similar to substrings of each long string. In particular, if the long strings are similar in terms of a distance that allows more general errors, the short string might be similar in terms of a distance that allows more specific errors.

Here "errors" includes mismatches, insertions, deletions, copies, duplications, moves, etc. "Specific errors" means a class composed of few of these, and "general errors" a class comprising many of these. The latter is observed frequently in genome sequences; two long genome sequences similar in the edit distance usually have substrings similar in the Hamming distance. For example, BLAST[1,2] and FASTA[9] that are de facto standard similarity search tools in bioinformatics, use this principle to find long similar strings; they find a pair of substring that are exactly the same, and check whether they are a part of similar long substrings. This principle motivates us to use short frequent substrings as seeds of long frequent substrings. That is, we first find short frequent substrings and then extend each short string in both directions until it becomes not similar to its occurrences. Accordingly, the problem for long strings is formulated as follows.

Problem: Cluster-based Long Frequent String Mining
input string set $\mathcal{D} = \{S_1, \ldots, S_m\}$
1. (Seed enumeration) Find groups (clusters) of substrings of short length in the set of strings such that the substrings in the group are similar to each other.

2. (Centroid computation) For each cluster, compute a string similar to all the substrings in a cluster, and extend the string in both directions until it is not similar to many substrings in the cluster.

If a seed is too short in length, it may be included in many different frequent string patterns. On the other hand, the seeds have to be shortened as much as possible to achieve the computational efficiency. This trade-off is a key to achieving practical efficiency with this model.

4 Algorithm

In this paper, we propose an efficient algorithm for solving these problems. The key idea is using a multi-sorting algorithm[14] for finding similar short substrings. This algorithm inputs a set of strings of the same length l, and finds all the string pairs having a Hamming distance of at most d. Applying this algorithm to all substrings in the string data gives a graph representation of similarity among all substrings. A number of clustering algorithms can be used to find clusters, but here we use a simple method for adapting large-scale data.

Hereafter, we assume that a substring has length l. For a substring of length l, let $N(S)$ be the set of substrings of length l of a string in the string data such that the Hamming distances to S is at most d. Note that S is included in $N(S)$. To make a cluster, we choose a substring S maximizing $|N(S)|$ which corresponds to a maximum degree vertex in the graph representing the similarity relation among the substrings. We then make a cluster by $N(S)$. We repeat this process to find more clusters, after removing the substrings in the cluster from the candidates of S. The process ends when $|N(S)|$ achieved a value of less than σ, that is a given threshold value.

For each cluster, we compute its representative string $C(N(S))$, which we call *centroid*, by using a majority voting algorithm described as follows. Here, $v(\{s_1, \ldots, s_k\}, \rho) = c$ if $k\rho$ letters s_j in $\{s_1, \ldots, s_k\}$ are c, and is the wildcard otherwise.

```
Algorithm. StringVote ($\{S_1, \ldots, S_k\}, \rho(> 0.5)$)
1. for $i := 1$ to $l$, set $S^*[i]$ to $v(\{S_1[i], \ldots, S_k[i]\}, \rho)$
2. return $S^*$
```

The length of $C(N(S))$ is l, and its ith letter is determined by the voting, i.e., the letter appearing most often in all the ith letters of strings in $N(S)$. For example, if the first letters of three strings in a cluster are all A, B, and A, the majority is A, and the first letter of the centroid $C(N(S))$ becomes A. For the case in which the majority letter is not especially significant, we use a threshold value ρ so that if the majority letter appears less than $\rho|N(S)|$ in the substrings in $N(S)$, we set the letter to the "wildcard", instead of the majority letter.

Instead of the voting, we can simply use S itself as the centroid. However, in some case, S can be different from the common string. For example, $S = BBAAAAAA$, and all strings in $N(S)$ are $AAAAAAAA$ except for one,

$BBBBAAAA$. The representative of this cluster should be $AAAAAAAA$, but S is different from it. However, in the analysis explained in the next section, choosing S has an advantage. Using the voting algorithm, our algorithm for Cluster-based Frequent String Mining problem is described as follows.

```
Algorithm. FreqString ($\mathcal{D}, l, d, \sigma, \rho$)
    $\mathcal{D}$: string set, $l$: substring length, $d$: Hamming distance threshold, $\sigma$: degree
    threshold, $\rho$: voting threshold
1. $\mathcal{S}$ := all substrings of length $l$ in a string in $\mathcal{D}$
2. call multi-sorting algorithm to compute $N(S)$ for all substrings in $\mathcal{S}$
3. choose a substring $S$ in $\mathcal{S}$ maximizing $|N(S)|$
4. if $|N(S)| < \sigma$ stop
5. compute $C(N(S))$ by StringVote ($N(S), \rho$)
6. $\mathcal{S} := \mathcal{S} \setminus N(S)$
7. if $\mathcal{S} \neq \emptyset$ go to 3
```

Let N denote the number of output pairs by the multi-sorting algorithm, i.e., $N = (\sum_{S \in \mathcal{S}} |N(S)|)/2$. In practical terms, the multi-sorting algorithm terminates in $O(l(|\mathcal{S}| + N))$ time if l is sufficiently large and d is sufficiently small, e.g., in the case where $l = 30$ and $d = 3$ for genome sequences[14]. Steps 1, 3, 4 and 7 can be done in $O(|\mathcal{S}|)$ time in total. Since step 6 can be done by removing all pairs including a substring in $N(S)$, and no pair is removed twice, thus accounting computation time for step 6 results $O(N)$. Step 5 can be done in $O(l\ |N(S)|)$ time, thus needs $O(lN)$ time in total. In summary, the computation time depends on the time for multi-sorting algorithm, and can be expected to be $O(l(|\mathcal{S}| + N))$ in practice. We can see this by looking at the result of our experiments; the memory usage that is linear to N and the computation time.

For finding longer frequent substrings, we extend the seeds as follows. Suppose that we obtain centroid S^* from a cluster $N(S)$. Then, for all substrings in the cluster, we compute the majority letter in the same way as given above, among all letters following the substrings in strings in $\mathcal{D}$. Note that if some substrings are located at the end of some texts, we consider that any following letter is a special letter '\$' meaning "void". The majority letter, or the wildcard, is the $l+1$ th letter of the extended centroid. We further compute the second following letter, the third following letter and so on. We want to stop this when the majority is often not significant. We stop the extension if there are more than θ wildcards in the last l letters of the extending centroid, where θ is a given threshold. The stopping criteria, majority threshold, and some other parameters or methods of the extension can be changed as the user likes, but to make the similarity of the strings uniform, we propose to use the same setting as that for the centroid computation and the multi-sorting algorithm. The algorithm is described as follows.

```
Algorithm. StringVoteExt ($S, \{p_1, \ldots, p_k\}, l, \rho(> 0.5), \theta$)
1. $i := 0$; $j := l - 1$; $S^* :=$ StringVote ($\{S[p_1, p_1 + l - 1], \ldots, S[p_k, p_k + l - 1]\}$,$\rho$)
2. $i := i - 1$; set $S^*[i + 1]$ to $v(\{S[p_1 + i], \ldots, S[p_k + i]\}, \rho)$
3. if $S^*[i + 1] \neq \$$ and $S^*[i + 1, i + l]$ has wildcards of at most $\theta$, go to 2.
```

```
4. $j := j + 1$; set $S^*[j + 1]$ to $v(\{S[p_1 + j], \dots, S[p_k + j]\}, \rho)$
5. **if** $S^*[j + 1] \neq \$$, and $S^*[j - l + 1, j + 1]$ has wildcards of at most $\theta$, **go to** 4.
6. **return** $S^*[i + 2, j]$
```

After computing the centroid, we remove all substrings in the cluster $N(S)$ from $\mathcal{S}$. We further remove the substrings from $\mathcal{S}$ that are included in the substring S' obtained by extending a substring in the cluster in both directions as the same length as the extent of the centroid. This is because in the next phase, the substrings removed would make a cluster, and the extension of the seed obtained by the cluster would be the same as that for the currently obtained centroid. The algorithm FreqLongString for Cluster-based Long Frequent String Mining is obtained by modifying steps 5 and 6 of FreqString as follows.

```
5. compute $C(N(S))$ by StringVoteExt ($\mathcal{D}$, { positions of $N(S)$ in $\mathcal{D}$}, $l$, $\rho, \theta$)
6. extend the substrings in $N(S)$ in the same length as for Step 5, and remove
all substrings included in the extended substrings from $\mathcal{S}$.
```

The additional computation time needed for this process is for the extension and the removal of the substrings. This needs, roughly, $O(Nh)$ time where h is the average length of the extension. Thus, not so many centroids are extended in long lengths, and therefore we may not lose computational efficiency.

To achieve efficiency of the computation, we are motivated to use exact matches as seeds, i.e., substrings having no mismatches, since finding exactly the same substrings is much easier than finding similar substrings. However, this may result in a loss of model efficiency. If we use identical substrings, the cluster size will be small, and the voting may not work well. If we use much shorter identical substrings, they are possibly included in other non-similar strings, thus the voting will be done for non-similar strings. Using similar substrings may exclude non-similar strings, and increase the cluster size.

5 Completeness of the Model

This section shows a completeness of the model, by stating that any frequently appearing string is similar to at least one output pattern. Suppose that $\mathcal{C}$ is the set of strings of length l obtained by FreqString, from a string set $\mathcal{D} = \{S_1, \dots, S_m\}$.

Lemma 1. *For any S and $N(S)$, the Hamming distance between S and $C(N(S))$ is less than d/ρ. In particular, it is less than $2d$ if $\rho > 1/2$.*

Proof. Let P be the set of mismatch positions of S and $C(N(S))$. For each p in P, let $Z(p)$ be the set of $S' \in N(S)$ such that at the position p, S' has a letter different from S. We then see that $|Z(p)| \geq \rho(|N(S)|)$. Since each $S' \in N(S)$ can have at most d positions mismatching S, $\sum_{p \in P} |Z(p)| \leq d(|N(S)| - 1)$. These imply that

$$d|N(S)| - d \geq \sum_{p \in P} |Z(p)| \geq \sum_{p \in P} \rho|N(S)| = |P|\rho|N(S)|,$$

thus we have $|P| < d/\rho$ □

Lemma 2. *Suppose that at least σ substrings in strings in $\mathcal{D}$ have Hamming distances at most $d/2$ to S. If S is a string in $\mathcal{S}$, there is always a string $S^* \in \mathcal{C}$ such that the Hamming distance between S and S^* is at most $(1 + 1/\rho)d$.*

Proof. From the assumption we have that $|N(S)| \geq \sigma$. Thus, S is included in $N(S')$ for some S' in Step 5 of algorithm FreqString. Since the Hamming distance between S and S' is at most d, and from Lemma 1, the Hamming distance between S' and $C(N(S'))$ is at most d/ρ, we have that the Hamming distance from S to $C(N(S'))$ is at most $(1+1/\rho)d$. $C(N(S'))$ is a member of $\mathcal{C}$, therefore the statement holds. □

Lemma 3. *Suppose that d is even and at least σ substrings in strings in $\mathcal{D}$ have Hamming distances at most $d/2$ to S. There is always a string $S^* \in \mathcal{C}$ such that the Hamming distance between S and S^* is at most $(1.5 + 1/\rho)d$.*

Proof. Let $\mathcal{H}$ be the set of substrings of strings in $\mathcal{D}$ such that the Hamming distance to S is at most $d/2$. Since $|\mathcal{H}| \geq \sigma$ and any two strings in $\mathcal{H}$ have Hamming distance of at most d, at least one of $\mathcal{H}$ belongs to $N(S')$ that induces a string $S^* \in \mathcal{C}$. From Lemma 1, the Hamming distance from S to S^* is at most $(1.5 + 1/\rho)d$. □

From these lemmas, we can see that the output of our algorithm covers all frequent strings. When we choose S as the centroid of $N(S)$, the upper bound in the statement of the lemma can be improved. From the proof of the above lemma, there is a string S' such that Hamming distance to S is at most $d/2$, and included in some cluster $N(S^*)$. Thus, the Hamming distance from S to S^* is bounded by d if S is a substring of a string in $\mathcal{D}$, and $1.5d$ otherwise.

6 Algorithm for Similar Short Substrings

This section is a brief summary of the multi-sorting algorithm[14]. The problem is formulated as follows.

Problem: Short Hamming Distance String Pair Enumeration
Input: A multiset $\mathcal{S}$ of strings of fixed length l, and threshold value d
Output: All pairs of strings S_1 and S_2 such that Hamming distance of S_1 and S_2 is at most d.

We first choose $k > d$, and then partition each string $S \in \mathcal{S}$ into k substrings (blocks) in the same way. The ith substring is called the ith block, and is denoted by $B(S, i)$. We denote by $\mathcal{K}$ the collection of subsets of size $k - d$ of $\{1, \ldots, k\}$, i.e., $\mathcal{K} = \{K \subseteq \{1, \ldots, k\} \mid |K| = k - d\}$. $sig(S, K)$ denotes the concatenation of blocks $B(S, i)$ of all $i \in K$. Observe that any two strings of Hamming distance at most d have at least $k - d$ blocks. The algorithm finds pairs of strings having $k - d$ same blocks as the candidates of the solutions, and the number of candidates is quite small compared to all possible pairs.

Table 1. Results for X chromosome; length of 1,000 letters

	d	200	400	800	1600	3200	6400	12800	25600	51200	102400	204800	291572
time	0	0.45	0.9	1.84	3.71	7.23	14.9	30	65	141	339	889	1559
(sec)	1	0.51	1.08	2.23	4.63	9.23	19.5	42	99	296	1197		
	2	0.58	1.3	3.09	7.22	15.4	34	76	203	804			
	3	0.91	2.05	5.18	13.1	28	66	155	456	2082			
memory	0	72	72	72	72	80	112	190	350	659	1284	1983	2705
(MB)	1	72	72	72	72	81	112	190	350	1217	3684		
	2	72	72	72	72	72	81	112	195	2879			
	3	72	72	72	72	72	89	126	316	5212			
#short	0	7	19	25	25	68	83	178	461	1603	4879	7003	15021
patterns	1	42	101	179	291	471	803	1696	5935	15836	41939		
	2	53	166	288	433	1168	1820	3770	14874	31193			
	3	88	415	1022	2537	1978	4239	6945	15879	58460			
#long	0	0	1	2	3	4	7	11	26	56	111	273	341
patterns	1	1	5	13	28	45	66	107	234	569	1019		
	2	2	4	20	38	52	136	229	493	980			
	3	3	7	29	55	64	130	254	349	398			
average	0	0	1123	667	429	301	335	220	218	193	244	304	277
length	1	87	302	205	158	142	142	142	173	180	204		
	2	106	432	191	167	161	133	130	146	152			
	3	126	278	161	122	121	130	113	123	219			

The framework of the algorithm is simple; for each $K \in \mathcal{K}$, we classify the strings in equivalence classes having the same $sig(S, K)$. This can be done by a radix sort like algorithm in linear time in the input size. Then, for each equivalence class, we find the pairs with Hamming distance of at most d by comparing all pairs of strings in the class. If the Hamming distance of S_1 and S_2 is at most d, they have at most d mismatches, thus have at least $k - d$ identical blocks. Thereby $sig(S_1, K) = sig(S_2, K)$ holds for some $K \in \mathcal{K}$. Hence, the pair is selected for output when the algorithm examines K.

If ${}_kC_d$ is small, and each block contains sufficiently many letters so that each equivalence class is small on average, the computation time will be short. Thus, practical high performance is for problems with low error rate, such as comparisons of genome sequences. The computational experiments reported in [14] showed that by setting $l = 30$ and $d = 2$, strings taken from anywhere within genome sequences of up to 10 million strings can be processed in few minutes.

7 Experiments

Our implementation is coded in C, and no sophisticated libraries such as binary trees are used. All experiments were done on a 3.2 GHz Core i7-960 computer with a Linux operating system having 24GB of RAM memory. Note that we did not use multi-cores in the experiment. The codes and the instances we used can be found available at the author's website
(`http://research.nii.ac.jp/~uno/codes.html`).

Table 2. Results for word sequence; length of 1,000 letters

	d	25	50	100	200	400	800	1600	3200	4700
time	0	0.1	0.19	0.46	1.26	2.93	22.9	43	175	267
(sec)	1	0.22	0.6	1.93	7.22	22.4	110	388	2978	
	2	1.72	6.23	27	120	466	2155			
memory	0	36	36	36	36	36	62	99	257	354
(MB)	1	36	36	36	72	52	126	330	1575	
	2	36	36	43	107	280	885			
#short	0	82	145	499	1905	3717	8673	20773	66915	86540
patterns	1	296	949	3660	12639	32825	92408	228485	734335	
	2	1898	4572	10242	21067	40413	85325			
#long	0	36	74	271	696	1409	3128	6417	16526	19665
patterns	1	72	168	339	762	1493	3226	6425	15571	
	2	89	177	347	715	1455	3224			
average	0	13	16	14	17	18	17	17	17	18
length	1	16	15	16	16	16	14	14	15	
	2	14	14	13	13	13	12			

Table 3. Results for change in pattern length (X chromosome); degree is average number of similar patterns, and deg< 2 is ratio of patterns that have at most two occurrences

l, d	10,0	15,0	15,1	15,2	20,0	20,1	20,2	30,0	30,1	30,2	30,3	50,0	50,2	50,4	50,6
time(sec)	690	37	192	1391	22	62	174	30	42	76	155	63	82	151	366
memory (MB)	1072	150	439	1551	97	181	340	112	112	112	197	121	124	124	106
#short patterns	2466	512	11915	164853	314	4168	6808	83	834	1975	4743	34	113	586	1404
#long patterns	1388	150	1011	770	86	302	0	27	107	229	254	2	11	88	106
average length	27	141	75	45	167	113	0	220	142	130	113	360	157	69	57
degree	0	5	0	0	11	5	0	8	14	13	9	1	1	90	95
deg< 2	98	37	69	89	22	25	0	72	13	11	17	0	0	0	0

We used a genome sequence and a word sequence as instances. The genome sequence was the human X chromosome taken from National Center for Biotechnology Information (NCBI) database. It was 143,733,266 letters in length, and comprised 4 kinds of letters and a special letter 'N' meaning the wildcard (except comment lines). The instances we generated from the X chromosome were doubled by attaching its reverse, since genome sequences are often similar to the reverse of subsequence of the other sequences. The word sequences were generated from the benchmark data named "20 Newsgroups" (`http://people.csail.mit.edu /jrennie/20Newsgroups/`). With the exception of dashes and periods, all symbols were removed from the data, and each word was assigned a unique ID. The word sequence was the sequence of ID's of the words in the data. The alphabet size of the word sequence was 132,859, and the length was 5,454,672.

Tables 1 and 2 show the experimental results for the performance in terms of the changes of the input size and d. Since the computation time and the

Table 4. Results for change in pattern length; word sequence

l, d	8,0	12,0	12,1	18,0	18,1	18,2	18,3	25,0	25,2	25,4	40,0	40,2	40,4	40,6
time(sec)	2573	43	387	17.6	32	121	872	14	33	148	27	36	65	2731
memory (MB)	1557	110	380	3398	60	138	558	61	62	69	63	63	64	64
#short patterns	9380	6041	5351	6031	2484	3419	3357	3262	1143	2263	3261	428	562	804
#long patterns	6585	6417	6425	2499	3799	3754	3845	983	1696	2277	425	533	731	804
average length	14	17	14	18	20	17	18	15	14	16	17	17	15	13
degree	0	0	0	0	0	0	0	0	0	0	0	0	0	0
deg< 2	100	100	100	98	99	100	100	99	99	97	100	100	96	100

memory usage does not differ much between FreqString and FreqLongString, we show those of FreqLongString. We set the pattern length l to 30 on Table 1 and 12 on Table 2, σ to 10, ρ to 1/2, θ to 2, and found the extended centroids for the cases of $d = 0, 1, 2, 3$, and $d = 0, 1, 2$, respectively. The instances were generated by taking substrings starting from the beginning, and were written in length of 1,000 letters. Tables 3 and 4 show the results obtained by the changes of the seed length l. We use the instances of length 6,400,000 for the X chromosome, and 1,600,000 for the word sequence.

The computation time was almost linear in the length except for large instances with large d, and even though the length is hundred millions, the computation time was comparatively short. For large scale data, the number of similar substrings was quite huge, thereby the computation time was long. In such cases, we should use much larger l and d, for not losing the efficiency as shown below. The number of patterns did not substantially explode, and the average pattern length was stable. In testing the appearance of the patterns we obtained in the genome sequence, we counted the substrings such that the edit distance to the pattern string was at most the 10% of the pattern length. Few patterns appeared few times, say 5–10 times, but other patterns appeared more than 30 times. For the increase of d, the increase in the pattern number and pattern length appeared to saturate at some point. This implies that in practice we do not have to care about large d.

We can find much more patterns by increasing d, especially for genome sequences, but too large d results in a long computation time. Thus, using small d would be a good solution. In genome sequences, we found many short patterns with small l, and few long patterns with large l. This is caused by that for small l, there are too many similar substrings for a substring. Such many substrings would be a part of non-similar long strings, thereby we could find only their common short substrings. For large l, we might not unify the groups, but when d is small, especially $d = 0$, we could find only few groups composed of quite similar substrings, and thus obtained very few patterns. By using slightly large d and non-short l, we can find many sufficiently long patterns within a quite short time. This result supports our motivation to use similar short substrings for frequent string mining. In particular, we can see many copies/quotations in the word sequences, and consequently we found many groups of copies for the case of $d = 0$, and the large average length of patterns. By introducing similarity,

Table 5. Results with using interleaving sampling; X chromosome

	d	3200	6400	12800	25600	51200	102400	204800	291572
time	0	2.53	5.23	10.7	22.6	47	102	246	386
(sec)	1	3.04	6.38	13.7	30	75	216	680	1316
	2	3.85	9.46	20.8	51	165	701	2550	
	3	6.21	13.9	33	92	374	2108		
#long	0	1	3	4	37	152	399	995	1313
patterns	1	31	48	72	218	563	1116	2566	3330
	2	46	115	193	454	1102	1454	4278	
	3	104	171	284	614	1052	1929		
average	0	1710	977	769	353	378	393	413	386
length	1	189	200	178	234	262	262	293	280
	2	160	164	161	215	215	217	227	
	3	139	135	136	181	193	211		

Table 6. Results with using interleaving sampling; X chromosome

	d	400	800	1600	3200	4700
time	0	1.21	10.9	18.4	56	93
(sec)	1	6.41	35	108	677	
	2	125	547	2191		
#long	0	777	1733	4881	13409	16655
patterns	1	1351	3095	6182	14798	
	2	1338	3164	5908		
average	0	16	4881	17	17	18
length	1	16	6182	15	15	
	2	15	5908	14		

we could find shorter patterns, that would not come from the groups of "copy and paste".

We also tested the similarity between patterns discovered. For each pattern S, we counted the number of strings S' similar to it; such that S (or S') had an edit distance of at most $0.1|S|$ to a substring of S'. We tested the output in the second experiment, and show results in the last two rows of Tables 3 and 4. The first row shows how many patterns are similar to one pattern on average, and the second row shows the fraction of patterns having at most two other similar patterns. In all cases, each pattern is similar to at most 15% of other patterns on average, and there are several patterns not similar to other patterns. The patterns we obtained from the data are in some sense independent to each other. In word sequences, and genome sequences with $d = 0$, almost all patterns have no similar pattern. It comes from that we found only groups of quite similar substrings, and "copy and paste", thus the strings in different groups are not similar to each other.

The computation time can be reduced by using "interleave sampling" of substrings proposed in [14]. By using the method, we can find continuous pairs of

Table 7. Comparison of the performance of the algorithms

algorithm	distance	errors	problem size	#patterns	CPU time
FreqLongString	edit distance	10%	100MB	1000	1000
(hill climbing)[8]	edit distance	2 or 3	>10MB	huge	1 day
HomologMiner[5]	Hamming distance	-	30MB	few	10 hours
RepeatScout[11]	edit distance	-	50MB	few	2 hours

substrings having small Hamming distance, i.e., we find pairs of substrings S_1 and S_2 such that $S_1[i, i+l]$ and $S_2[i, i+l]$ have a Hamming distance of at most d for any i. When we find such string pairs from short string pairs, we never miss sufficiently long continuously similar substrings, even if we take only $1/c$ of all substrings, for some constant c such as $c = l^{1/2}$. This enables us to fasten the computation without losing the efficiency. In our experiments shown in Table 5 and Table 6, we could reduce the computation time to 1/5, without decreasing the number of patterns found, and the length of the patterns.

Comparison with Other Methods
The comparison of the methods for this problem is actually very difficult, since they share the same task, but the problems formulations are different. For example, many algorithms in bioinformatics finds the frequent string from exact matches. They are faster than those of approximate matches, and the accuracy and completeness might be less, but no one can be sure about this. Some algorithms reduce the input data by domain specific background knowledge, and/or some corpus for efficient computation. It is hard to have a good criteria for the evaluation of algorithms. Instead of a good evaluation criteria, we just show the results of the computational experiments in Table 7, that are reported in literature. The numbers are approximations.

8 Conclusion

We considered the need to develop more practical models for frequent strings with approximate matches. We observed the motivation and the structures of the problem, and formulated the problem via clustering; first find all the similarity relations between the substrings of the given database, enumerate all clusters, and compute a representative string for each cluster. An algorithm we proposed to extend the short frequent string helps us to obtain much longer frequent strings in a short time. By using the multi-sorting algorithm proposed by the authors for the similarity computation, our algorithm could perform quite well even for large-scale real-world string data. The number of patterns we obtained in this approach was not excessively large such that the patterns obtained are tractable. The experimental results showed that by introducing the similarity, we can obtain much more long patterns or non-trivial patterns without drastically increasing the computational cost. We used Hamming distance for the similarity evaluation, but the obtained patterns are actually frequent even in the term of the edit distance. It is noteworthy that using simple criteria makes it possible to find interesting, large patterns with more sophisticated criteria. Interesting future research is to extend this algorithm to handle more sequence-like string patterns to accept more ambiguous data such as natural language texts.

Acknowledgments. We gratefully thank to Prof. Hiroki Arimura of Hokkaido University, and Prof. Takashi Washio of Osaka University for their valuable comments. This research is supported by JST PRESTO and Funding Program for World-Leading Innovative R&D on Science and Technology, Japan.

References

1. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. Journal on Molecular Biology 215, 403–410 (1990)
2. Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Research 25, 3389–3402 (1997)
3. Hébert, C., Crémilleux, B.: Mining Frequent δ-Free Patterns in Large Databases. In: Hoffmann, A., Motoda, H., Scheffer, T. (eds.) DS 2005. LNCS (LNAI), vol. 3735, pp. 124–136. Springer, Heidelberg (2005)
4. Goethals, B.: The FIMI repository (2003), `http://fimi.ua.ac.be/`
5. Hou, M., Berman, P., Hsu, C.H., Harriset, R.S.: HomologMiner: Looking for Homologous Genomic Groups in Whole Genomes. Bioinformatics 23, 917–925 (2007)
6. 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)
7. Manber, U., Myers, G.: Suffix Arrays: A New Method for On-line String Searches. SIAM J. on Comp. 22, 935–948 (1993)
8. Mitasiunaite, I., Boulicaut, J.-F.: Introducing Softness into Inductive Queries on String Databases. In: Databases and Information Systems IV, pp. 117–132. IOS Press (2007)
9. Pearson, W.R.: Flexible sequence similarity searching with the FASTA3 program package. Methods in Molecular Biology 132, 185–219 (2000)
10. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.-C.: PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In: ICDE 2001, pp. 215–224 (2001)
11. Price, A.L., Jones, N.C., Pevzner, P.A.: De novo Identification of Repeat Families in Large Genomes. Bioinformatics 21(suppl. 1), 351–358 (2005)
12. Roth, F.P., Hughes, J.D., Estep, P.W., Church, G.M.: Finding DNA Regulatory Motifs within Unaligned Noncoding Sequences Clustered by Whole-genome mRNA Quantitation. Nature Biotechnology 16, 939–945 (1998)
13. Saha, S., Bridges, S., Magbanua, Z.V., Peterson, D.G.: Computational Approaches and Tools Used in Identification of Dispersed Repetitive DNA Sequences. Tropical Plant Biol. (2008), doi:10.1007/s12042-007-9007-5
14. Uno, T.: Multi-sorting Algorithm for Finding Pairs of Similar Short Substrings from Large-scale String Data. Knowledge and Information System 25, 229–251 (2010)
15. Wang, J., Han, J.: BIDE: Efficient Mining of Frequent Closed Sequences. In: ICDE 2004, pp. 79–90 (2004)

OntoDM-KDD: Ontology for Representing the Knowledge Discovery Process

Panče Panov[1], Larisa Soldatova[2], and Sašo Džeroski[1]

[1] Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia
{pance.panov,saso.dzeroski}@ijs.si
[2] Brunel University, London, United Kingdom
larisa.soldatova@brunel.ac.uk

Abstract. In this article, we present an ontology for representing the knowledge discovery (KD) process based on the CRISP-DM process model (OntoDM-KDD). OntoDM-KDD defines the most essential entities for describing data mining investigations in the context of KD in a two-layered ontological structure. The ontology is aligned and reuses state-of-the-art resources for representing scientific investigations, such as Information Artifact Ontology (IAO) and Ontology of Biomedical Investigations (OBI). It provides a taxonomy of KD specific actions, processes and specifications of inputs and outputs. OntoDM-KDD supports the annotation of DM investigations in application domains. The ontology has been thoroughly assessed following the best practices in ontology engineering, is fully interoperable with many domain resources and easily extensible. OntoDM-KDD is available at `http://www.ontodm.com`.

Keywords: Knowledge Discovery in Databases, CRISP-DM, Data Mining Investigation, Data Mining, Domain Ontology.

1 Introduction

Recent surveys of research challenges for knowledge discovery in databases (KDD) and data mining (DM) list the mining of different types of structured data in a uniform fashion, the use of domain knowledge, and the support for complex KDD processes as the top-most open issues in the domain [1–3]. Much of the research in recent years has also focused on the automation and overall support of the KDD process. This involves development of standards for performing the KDD process as well as formal representations of the processes in the form of workflows [4, 8, 10, 13]. Specific issues addressed include methods that automate the composition of data mining operations into executable workflows. Finally, providing a mechanism for recording of results and the experimental settings of the DM experiments obtained by executing the workflows on sets of data is becoming important for ensuring the repeatability and reuse of results [14].

One of the most prominent proposals for standardizing the process of knowledge discovery in the context of representing and performing data mining investigations is the Cross Industry Standard Process for Data Mining (CRISP-DM) [4].

It is a process model that describes data mining investigations performed in practical applications. The CRISP-DM process model is based on commonly used approaches that expert data miners use to tackle and solve the practical problems in the domain.

The formalization of scientific investigations has also proved to be a prominent area of research. It includes providing a formal representation of objects and processes involved in scientific investigations in a knowledge representation framework, such as terminologies, taxonomies, and ontologies. The largest developments in this sense have taken place in biological and biomedical domains (e.g., the Robot Scientist project[5]). In addition, the state-of-the-art also includes initiatives to support and unify the representational mechanisms for recording scientific investigations under a single framework (e.g., the Open Biomedical Ontologies (OBO) Foundry [6]).

In the domain of KDD and DM, there exist several proposals of domain ontologies but the majority of them are light-weight application oriented ontologies aimed at covering a particular use-case in data mining. Initial systems that include ontologies are used to systematically describe the processes in machine learning and DM (e.g., the IDA system [7]). Next, there are ontology developments aimed to support workflow composition and planing of workflows [7–10], support of data mining applications on the GRID [11, 12], and support of meta-learning and meta-mining [13]. Finally, there are ontologies designed to support machine learning experiments in the context of experiment databases [14].

In our previous work [15, 16], we formally represented and described the complex domain of data mining by developing OntoDM, a general-purpose domain ontology of DM that takes into account the state-of-the-art developments in the area of formalization of scientific investigations. In this paper, we present the OntoDM-KDD ontology, a novel sub-ontology module of OntoDM. OntoDM-KDD (v.1) introduces the data mining investigations as a representational mechanism to describe the complete process of KDD, based on the CRISP-DM process model. The ontology includes a taxonomy of KD specific processes, actions and representations of inputs and outputs. Finally, the ontology has been thoroughly assessed following the best practices in ontology engineering, evaluated and validated by the applications on a use case.

2 Design

The OntoDM-KDD ontology is based on the CRISP-DM process model [4]. Its main goal is to be general enough to allow the representation of knowledge discovery processes and data mining investigations performed in practical applications. Based on this main goal, we identified a list of competency questions that our ontology is designed to answer.

Table 1. Examples of OntoDM-KDD competency questions

What is the description of a DM investigation X?
What is the set of publications about the investigation for a DM investigation X?
What is the DM investigation that is reported by a publication X?
What is the set of actions realized by the KD phase X for a DM investigation Y?
What is the set of DM investigations that realize an action X in a KD phase Y?
What is the process that precedes process X in KD phase Y for a DM investigation Z?

Examples of OntoDM-KDD competency questions are listed in Table 1. From the list of questions, we can see that the ontology need to include basic information entities for representing data mining investigations, such as action specifications, reports, and textual entities. Furthermore, the ontology need to contain processual entities that are parts of the knowledge discovery process.

In order to ensure the interoperability of OntoDM-KDD with other resources, the OntoDM-KDD ontology follows the OBO Foundry design principles [1]. These include, for example, the use of an upper level ontology, the use of formal ontological relations, single inheritance, and the re-use of already existing ontological resources where possible [6]. The use of these design principles enables cross-domain reasoning, facilitates wide reusability of the developed ontology, and avoids duplication of ontology development efforts.

OntoDM-KDD imports the upper level classes from the Basic Formal Ontology (BFO1.1)[2] and formal relations from the OBO Relational Ontology (RO)[3] [17] and uses an extended set of RO relations. BFO an RO were chosen as they are widely accepted, especially in the biomedical domain. Following best practices in ontology development, the OntoDM-KDD ontology reuses appropriate classes from a set of ontologies, that act as mid-level ontologies. These include the Ontology for Biomedical Investigations (OBI)[4], the Information Artifact Ontology (IAO)[5], and the Software Ontology (SWO)[6]. Classes that are referenced and reused in OntoDM-KDD are imported by using the Minimum Information to Reference an External Ontology Term (MIREOT) principle [18] and extracted using the OntoFox web service[7].

OntoDM-KDD is expressed in OWL-DL[8], a de facto standard for representing ontologies. The ontology is being developed using the Protege[9] ontology editor. The ontology is freely available at `http://www.ontodm.com` and at Bio Portal[10].

[1] OBO Foundry: `http://obofoundry.org/crit.shtml`
[2] BFO: `http://www.ifomis.org/bfo`
[3] RO: `http://purl.org/obo/owl/OBO_REL`
[4] OBI: `http://obi-ontology.org/page/Main_Page`
[5] IAO: `http://code.google.com/p/information-artifact-ontology`
[6] SWO: `http://theswo.sourceforge.net`
[7] OntoFox: `http://ontofox.hegroup.org`
[8] OWL-DL: `http://www.w3.org/TR/owl-guide`
[9] Protege: `http://protege.stanford.edu`
[10] BioPortal: `http://bioportal.bioontology.org`

3 The Structure of OntoDM-KDD

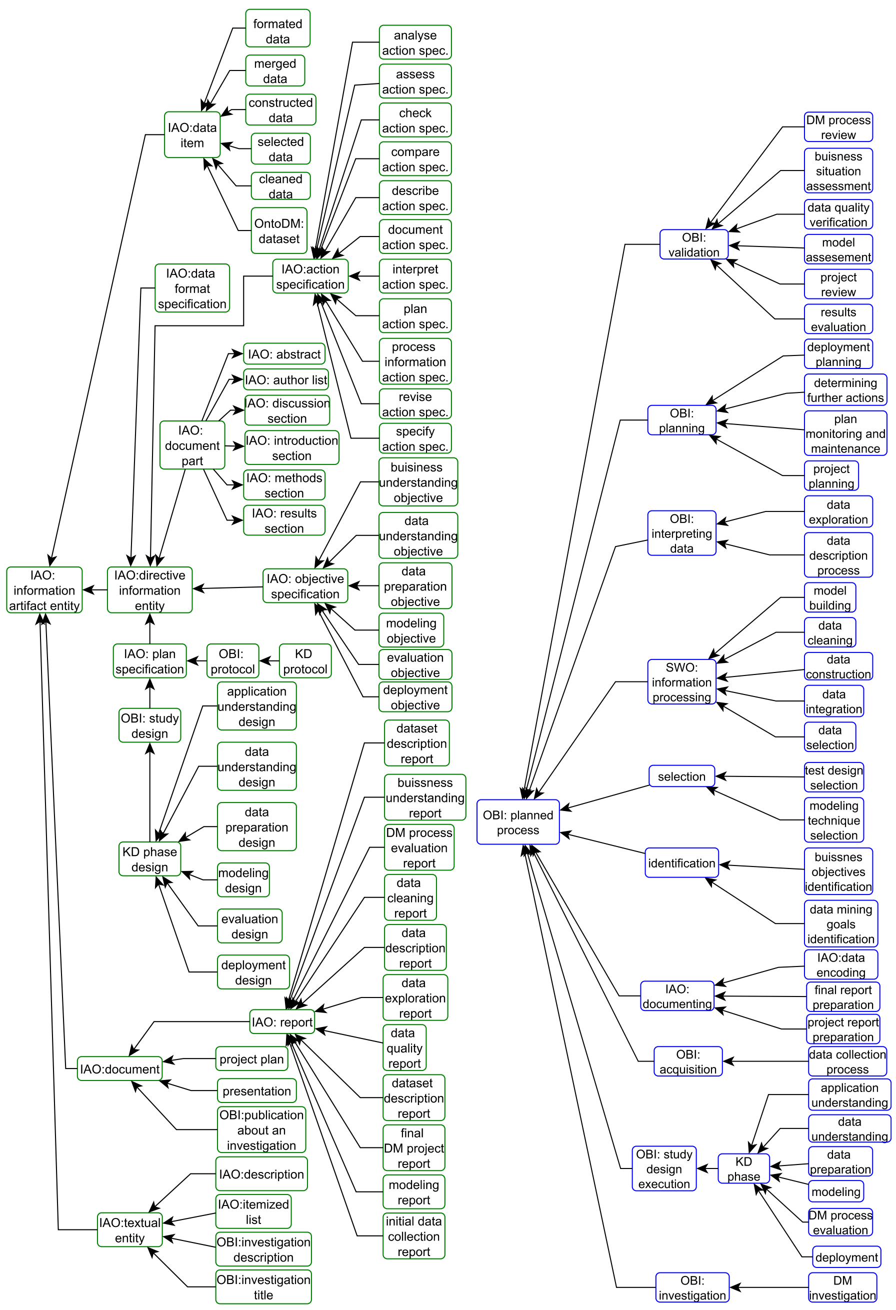
The CRISP-DM process model, at the top level, is organized into six phases: business understanding phase, data understanding phase, data preparation phase, modeling phase, evaluation phase, and deployment phase [4]. It defines the outputs of each CRISP-DM phase and the second-level generic tasks. For example, the data understanding phase consists of four generic tasks: collect initial data, describe data, explore data, and verify data quality. The level of specialized tasks (third level) describes how the generic tasks should be carried out in specific situations, in terms of activities. For example, the describe data task includes activities for volumetric analysis of data, assessment of the attribute types and values, etc. The fourth level, the level of process instances, describes the actions, decisions and results of an actual data mining investigation performed in the domain of interest.

For the purpose of representing data mining investigations, it is very important to have the ability to represent entities that deal with information, such as data, documents, reports, models, algorithms, protocols, etc. We thus incorporate and further extend some classes of the IAO ontology. The IAO ontology is a mid-level ontology describing information content entities (e.g., documents), processes that consume or produce information content entities (e.g., documenting), material bearers of information (e.g., journals), and relations in which one of the relata is an information content entity (e.g., is-about).

Another important representational aspect is representation of processes. In OntoDM-KDD, we use and further extend classes from the OBI ontology, such as the OBI process taxonomy, which includes general processes such as documenting, planing, validation, etc. The OBI ontology aims to provide a standard for the representation of biological and biomedical investigations. It supports consistent annotation of biomedical investigations regardless of the particular field of study and is fully compliant with the existing formalisms in biomedical domains [19]. In addition, OBI defines an investigation as a process with several parts, including the planning of an overall study design, executing the designed study, and documenting the results. Finally, in OntoDM-KDD we include the SWO class *Information Processing* that represents processes in which input information is analysed or transformed in order to produce an output information.

In OntoDM-KDD we distinguish two description layers based on the mid-level ontologies that it extends (Fig.1). The first layer is the specification layer, that deals with information entities needed to describe and represent the DM investigations. The second layer is the application layer that deals with processual entities in order to represent processes that occur in a DM investigation.

The specification layer (Fig. 1(a)) consists of classes that are extensions of the IAO class *Information Content Entity*. At the top level, it includes classes such as *Data Item*, *Directive Information Entity*, *Document*, *Document Part* and *Textual Entity*. The *Directive Information Entity* class is further extended with *Action Specification*, *Data Format Specification*, *Objective Specification*, and *Plan Specification*. In addition, we also reuse the *Study Design* and *Protocol* classes from the OBI ontology.



(a) Specification layer (b) Application layer

Fig. 1. The structure of the OntoDM-KDD ontology. The imported classes include in their name the source ontology label (IAO, OBI, SWO, OntoDM) as a prefix, while the native OntoDM-KDD classes are shown without such a label.

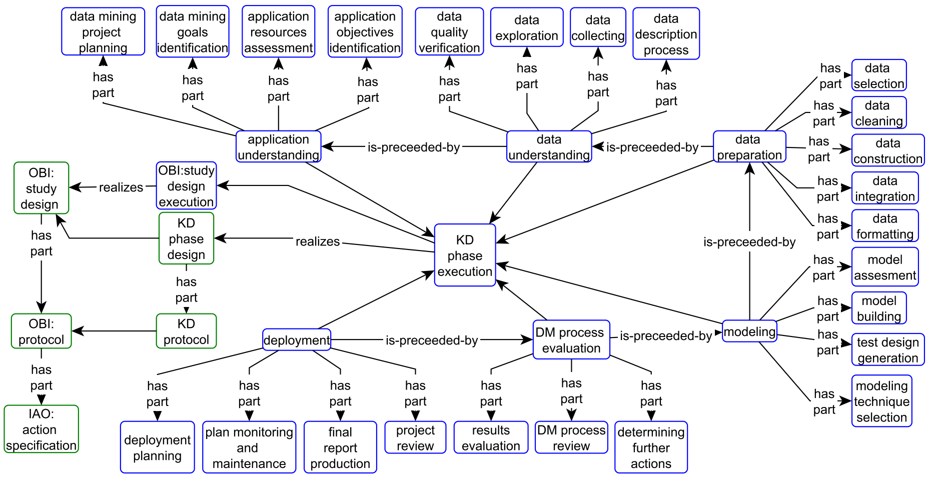


Fig. 2. KD phase execution and the *has-part* taxonomy of KD specific processes. The unlabeled arrows represent *is-a* relations. Classes represented with green boxes belong to the specification layer, while the blue boxes belong to the application layer.

The application layer (Fig. 1(b)) consists of classes that are extensions of the OBI class *Planned Process*. These include general classes of processes such as: *Validation*, *Planning*, *Interpreting Data*, *Information Processing*, *Selection*, *Identification*, *Documenting*, and *Acquisition*. Finally, the application layer includes the OBI *Investigation* class, which we further extend to define and represent a *Data Mining Investigation*.

4 Mapping the CRISP-DM Model to OntoDM-KDD

The phases level from the CRISP-DM model is represented in OntoDM-KDD with two aspects. In the specification layer, we represent the specification of the phases with the *KD Phase Design* class (Fig 2). It is a subclass of OBI *Study Design* which is comprised of *Protocols*. For example, the *Data Understanding Design* contains the specification of the data understanding phase.

The *KD Phase Design* is realized during *KD Phase Execution*. *KD Phase Execution* is represented as a processual entity in the application layer and is extended with KD specific phases from the CRISP-DM process model (Fig 2). These include *Application Understanding*, *Data Understanding*, *Data Preparation*, *Modeling*, *DM Process Evaluation*, and *Deployment* class. For example, *Data Preparation* process is a *KD Phase Execution* and realizes the *Data Preparation Design*. In this version of OntoDM-KDD, we can represent a sequential ordering of the KD phases by using the *is-preceded-by* relation (Fig.2).

Similar as the phases, the generic tasks from each phase from the CRISP-DM model are represented in OntoDM-KDD with two aspects. In the specification layer, we present specification of the tasks with the *KD Protocol Class*. For example, the *Data Understanding Design* contains as parts the *Data Collecting Protocol*, the *Data Describing Protocol*, the *Data Exploring Protocol* and the *Data Quality Verification Protocol*.

The executions of the protocols are represented in the application layer, and are parts of the *KD Plase Execution* process. For example, *Data Understanding* contains as parts sub-processes: *Data Collecting*, *Data Exploration*, *Data Description Process*, and *Data Quality Verification*. Each of the sub-processes is a sub-class of a more general processes class. For example, *Data Collecting* is a sub-class of *Aquisition* process.

The activities from the specialized tasks level of the CRISP-DM model are represented in OntoDM-KDD as actions. One of the most important parts of the specification layer is the taxonomy of KDD specific actions, represented by the extension of the *Action Specification* class. The action specification defines the actions that are realized in the processes. At the first level, we have the more general actions such as *Analyze Action*, *Assess Action*, *Check Action*, *Compare Action*, *Describe Action*, *Document Action*, *Interpret Action*, *Plan Action*, *Process Information Action*, *Revise Action*, and *Specify Action*. At the second level, the general actions are extended with KDD specific actions. Finally, each *KD Protocol* contains a set of action specifications as parts. For example, the *Data Exploring Protocol* includes the *Explore Data Action* and *Formulate Hypothesis Action*.

An *Investigation* is a planned process and includes the *Planning*, *Documenting*, and *Study Design Execution* processes (Fig. 3). Furthermore, an investigation is described with an *Investigation Title*, *Investigation Description*, and *Investigation Identifier*. In addition, the investigation produces as output a *Conclusion Textual Entity*. Finally, a *Publication About an Investigation* is a document about it and it is an output by the documenting sub-process. OntoDM-KDD defines a *DM Investigation* class as an extension of the OBI *Investigation* class (Fig. 3). A *DM Investigation* has as its part the *KD Phase Execution* process.

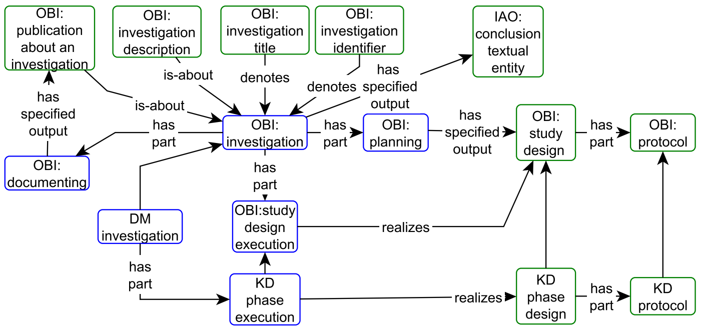


Fig. 3. The data mining investigation class in OntoDM-KDD

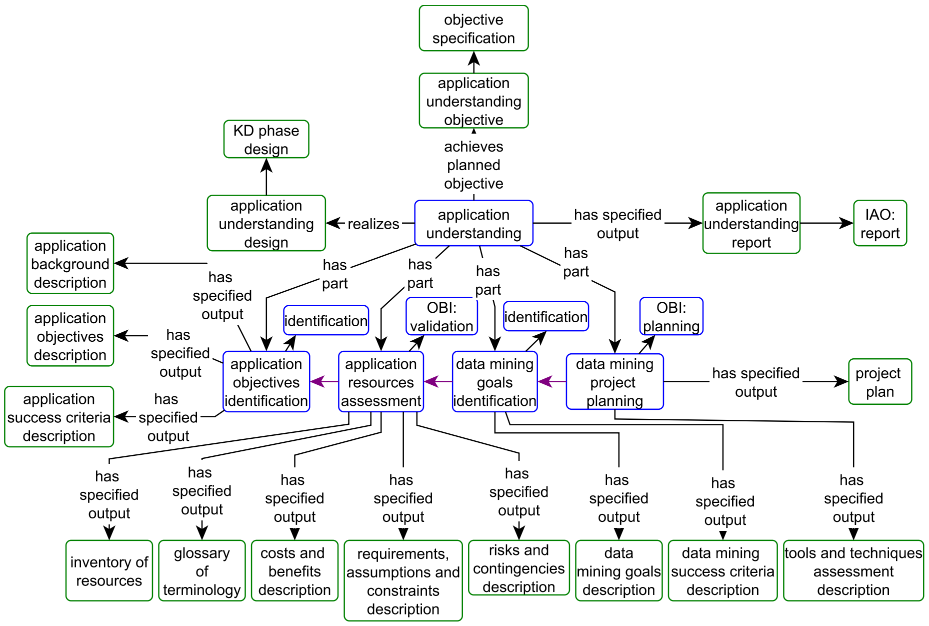

objective specification
application understanding objective
achieves planned objective
KD phase design
application understanding design
realizes
application understanding
has specified output
application understanding report
IAO: report
application background description
application objectives description
application success criteria description
has specified output
has part
identification
OBI: validation
identification
OBI: planning
application objectives identification
application resources assessment
data mining goals identification
data mining project planning
project plan
inventory of resources
glossary of terminology
costs and benefits description
requirements, assumptions and constraints description
risks and contingencies description
data mining goals description
data mining success criteria description
tools and techniques assessment description

Fig. 4. Application understanding process in OntoDM-KDD. The unlabeled arrows represent *is-a* relations, while coloured arrows represent *is-preceded-by* relation.

5 Example: Application Understanding Process

In this section, we present an example of representation of one phase from CRISP-DM in OntoDM-KDD. The initial phase in a DM investigation focuses on identifying the objectives and requirements of the investigation, from an application (or business) perspective. In CRISP-DM, this phase was named business understanding, while in OntoDM-KDD we generalize it as application understanding. The goal is to convert the knowledge about the application domain into a data mining problem definition and to generate a plan for achieving the application objectives.

The *Application Understanding* class is a sub-class of *KD Phase* and represents a planned process (Fig. 4). In the ontological vocabulary, the *Application Understanding* process can be defined as a *KD Phase* that realizes an *Application Understanding Design*, achieves the planned objective an *Application Understanding Objective* and has specified output *Application Understanding Report*. The *Application Understanding Process* includes as parts four sub-processes: *Application Objectives Identification*, *Application Resources Assessment*, *Identification of Data Mining Goals* and *Generation of a Project Plan.*

The process of *Application Objectives Identification* is a sub-class of the more general class of *Identification* processes. In this process, a data analyst (active participant or agent) needs to identify in detail, from the application (or business) perspective, what are the objectives to be achieved by applying DM to

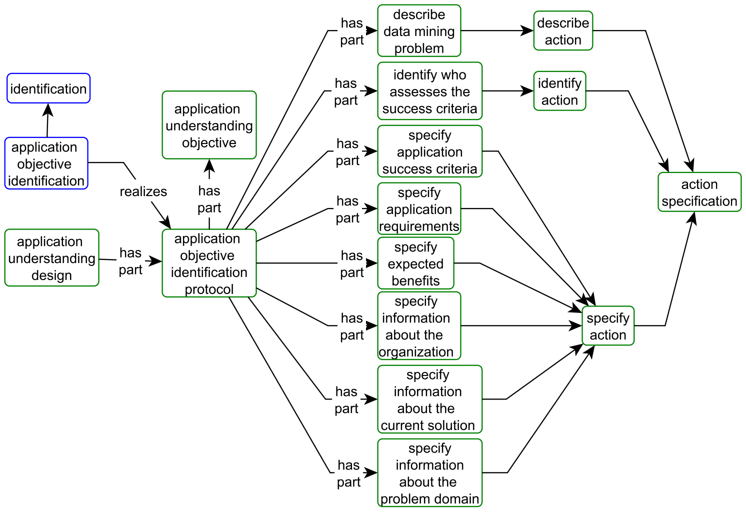

identification
application objective identification
realizes
application understanding objective
has part
application understanding design
has part
application objective identification protocol
has part
describe data mining problem
describe action
identify who assesses the success criteria
identify action
specify application success criteria
specify application requirements
specify expected benefits
specify information about the organization
specify information about the current solution
specify information about the problem domain
action specification
specify action

Fig. 5. The process of objective identification in OntoDM-KDD

the application domain at hand. At the end of the process, the analyst needs to produce as output an *Application Background Description*, an *Application Objectives Description*, and an *Application Success Criteria Description*.

The process of *Application Objective Identification* is a realization of an *Application Objective Identification Protocol* (Fig. 5). The protocol, which is a part of the *Application Understanding Design*, contains a specification of the actions that are realized in the process, such as *Describe Data Mining Problem*, *Specify Application Success Criteria*, *Identify who Assesses the Success Criteria* and others. These action specifications are subclasses of general classes of actions, such as *Describe Action*, *Specify Action* and *Identify Action*. In the OntoDM-KDD, we represent and provide action specifications for all processes.

The process of *Application Resources Assessment* is a sub-class of the OBI *Validation* process. It involves assessing the information about all resources, constraints, assumptions and other factors that need to be considered in order to determine the data mining goals and the project plan. The outputs of this process include: an *Inventory of Resources*, a *Glossary of Terminology*, *Costs and Benefits Description*, a *Requirements Assumptions and Constraints Description*, and a *Risks and Contingencies Description*.

The process of *Identification of Data Mining Goals* is a sub-class of the *Identification* process. The objective of this process is to produce a specification of the data mining goals and establish a set of data mining success criteria which can be used to evaluate the success of the data mining investigation at hand. The output of this process includes a *Data Mining Goals Description* and a *Data Mining Success Criteria Description*.

The process of *Data Mining Project Planning* is a sub-class of the OBI *Planning* process. The objective of this process is to produce a specification of a plan in order to achieve the data mining and application goals. The outputs of the process include a *Project Plan* and a *Tools and Techniques Assessment Description*.

6 Evaluation

We assess the quality of OntoDM-KDD from three different evaluation aspects. First, we analyze a set of ontology metrics. Then, we assess how well the ontology meets a set of predefined design criteria and ontology best practices. Finally, we assess how the ontology meets a set of predefined competency questions.

A variety of ontology metrics is available for assessing ontologies. We use the statistical ontology metrics from the Protégé software and the BioPortal web service (Tab. 2(a)). OntoDM-KDD has 264 classes, 34 relations and 2091 axioms. The size of the ontology is comparable with the average size of the OBO Foundry ontologies and the complexity (the number of relations and axioms) is higher than the average.

Table 2. OntoDM-KDD Evaluation

(a) Statistical metrics

Axiom count	2091
Class count	264
Individual count	0
DL expresivity	SHI
SubClassOf axiom count	521
DisjointClasses axiom count	53
Relations count	34
Annotation axioms count	1178

(b) An example of a competency question formalized in SPARQL-DL

What actions are realized by the KD phase X for the investigation Y?
Q(act):-Type(?act,action_specification), PropertyValue(?prot,has-part,?act), Type(?prot,protocol) PropertyValue(?kddphdesign,has-part,?prot), Type(?kddphdesign,kdd_phase_design), PropertyValue(?kddphplan, is-concretization-of, ?kddphdesign), PropertyValue(?x,realizes,?kddphplan), Type(x,kdd_phase_execution), PropertyValue(y,has-part,x), Type(y,investigation).

The ontology has been constructed following the best in ontology engineering and design criteria. The set of design principles (in total 29) is divided into four groups: scope and structural assessment; naming and vocabulary assessment; documentation and collaboration assessment; and availability, maintenance, and use assessment. The results of the evaluation are summarized on the ontology web page (`www.ontodm.com`). In sum, the design principles were closely followed during the development of OntoDM-KDD.

Following the recommendations by Gruniger and Fox [20], we first defined the ontology's requirements in the form of competency questions that the ontology must be able to answer (see above). Furthermore, having defined the language of the ontology, the competency questions are defined formally as an entailment with respect to the axioms in the ontology. In this way, one can evaluate the ontology and claim that it is adequate if the questions can be formulated in the language of the ontology. For that purpose, we formulated the questions using SPARQL-DL query language[11] [21] for querying OWL-DL ontologies. SPARQL-DL is a subset of the SPARQL language[12]. In Tab. 2(b), we show an example of an OntoDM-KDD competency question formulated in SPARQL-DL.

7 Usecase: Annotation of Data Mining Investigations

In this section, we present an example of how OntoDM-KDD can be used to annotate DM investigations in application domains. For this purpose, we focus on a DM investigation titled "Estimating forest properties from remotely sensed data using DM", published in a journal article by Stojanova et al. [22].

The DM investigation aimed at modeling forest properties, such as vegetation height and canopy cover, from remotely sensed data, by using DM algorithms. The final goal of this investigation was to use the models of the properties to generate forest maps that can be deployed in forest management and forest decision support systems. The DM investigation included: the study of the application domain; collection of data; preparation of the data; modeling of the forest properties; evaluation of the modeling process and deciding on the best model; generation of the forest property maps; and finally deployment of the generated maps in forest management systems.

In Fig. 6, we present a part of the annotation of this investigation. First, we define *dm investigation*[13] as an instance of the *DM Investigation* class that has as parts instances of the *planning*, *documenting*, and a *kd phase execution* processes (or their child classes). In addition, *'Estimating forest properties from remotely sensed data' denotes* the investigation and represents its title and the *investigation description* instance that *is-about* the investigation.

The documentation process *has-specified-output* a *Publication About An Investigation*, which represents an entity that *is-about* an investigation. An instance of this class is used to represent the journal article. In addition, this instance has as parts document part instances, such as *abstract*, *author list*, *institution list*, *introduction to a publication about an investigation*, *methods section*, *results section*, *discussion section of a publication about an investigation*, *conclusion to a publication about an investigation*, and *references section.* Finally, *'Estimating vegetation height and canopy cover from remotely sensed data with machine learning' denotes* the publication's title.

[11] SPARQL-DL: `www.w3.org/2001/sw/wiki/SPARQL-DL`
[12] SPARQL: `http://www.w3.org/TR/rdf-sparql-query/`
[13] Notation: with non-capitalized italics we denote instances of classes.

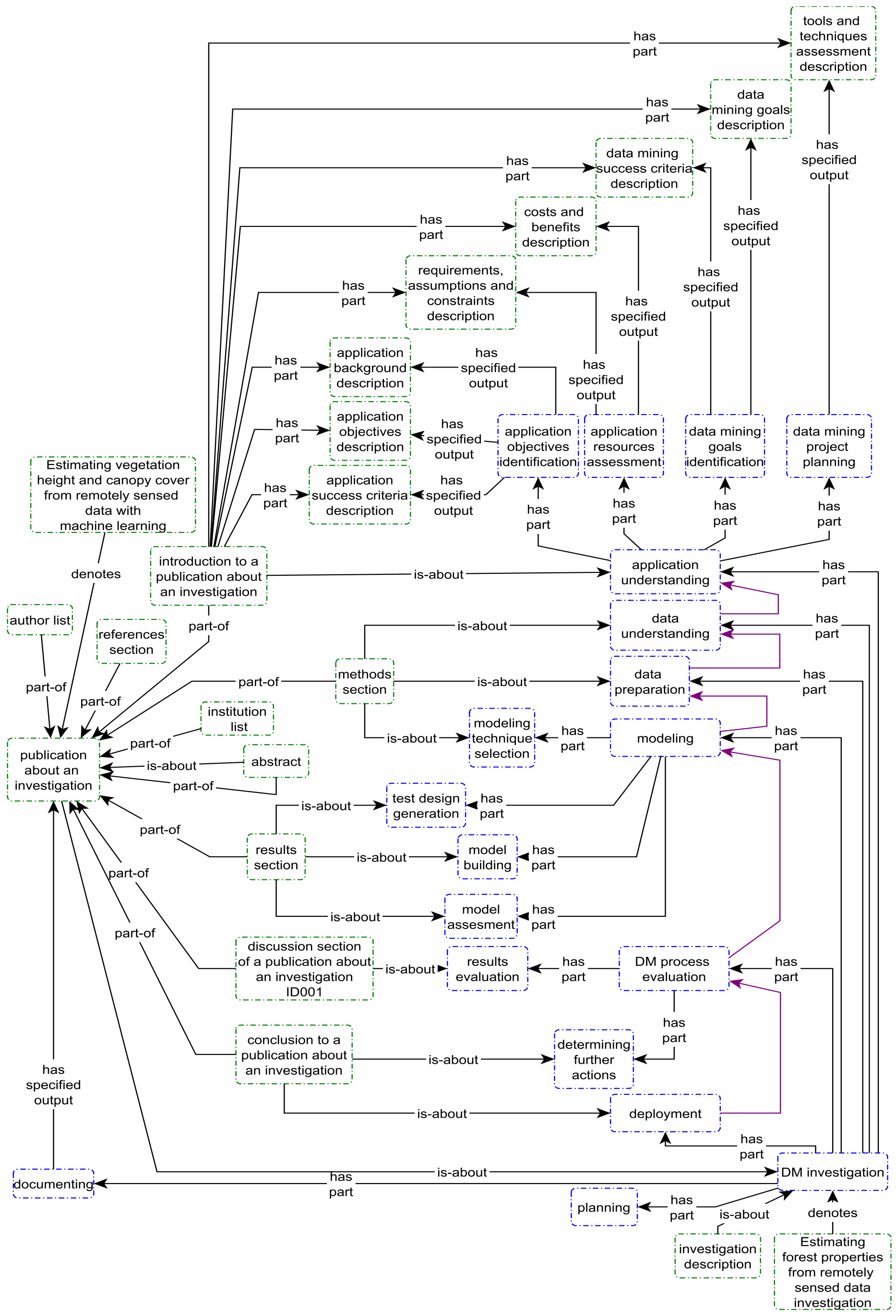

tools and techniques assessment description
data mining goals description
data mining success criteria description
costs and benefits description
requirements, assumptions and constraints description
application background description
application objectives description
application success criteria description
application objectives identification
application resources assessment
data mining goals identification
data mining project planning
Estimating vegetation height and canopy cover from remotely sensed data with machine learning
introduction to a publication about an investigation
application understanding
data understanding
data preparation
modeling
methods section
modeling technique selection
author list
references section
institution list
abstract
publication about an investigation
test design generation
results section
model building
model assesment
discussion section of a publication about an investigation ID001
results evaluation
DM process evaluation
conclusion to a publication about an investigation
determining further actions
deployment
documenting
DM investigation
planning
investigation description
Estimating forest properties from remotely sensed data investigation
has part
has specified output
is-about
part-of
denotes

Fig. 6. Part of an annotation of a DM investigation summarized in a journal article with terms from the OntoDM ontology

The introduction part of the article, represented by the *introduction to a publication about an investigation* instance, *is-about* the application understanding process. It has as parts descriptions that are outputs of processes that compose the application understanding process. For example, the introduction includes description instances such as *application background description*, *application objectives description*, and *application success criteria description*. These descriptions are instances of the *Textual Entity* class and are outputs of the *application objective identification* process instance. The same holds also for the other process instances that compose the application understanding process.

The methods section, represented by the *methods section* instance, contains parts that are about the data understanding process, the data preparation process and modeling technique selection (as a sub-process of modeling). More specifically, it contains description of the study area (the Kras region), data sources (LiDAR and Landsat), data descriptions (descriptive and output variables), and description of the DM techniques to be used. These descriptions are outputs of the the sub-processes of data understanding and data preparation, and the modeling technique selection process [14].

The results section, represented by the *results section* instance, contains parts that are about test design generation process, the model building process, and the model assessment process. These are all sub-processes of modeling. More specifically, it contains descriptions of the experimental design, DM algorithms applied, evaluation procedure and results (best models in terms of predictive performance and maps of vegetation height and canopy cover for the best model). These descriptions are outputs of the sub-processes of test design generation, model building and model assessment.

The discussion section, represented by the *discussion section of a publication about an investigation* instance, contains parts that are about a results evaluation process, a sub-process of a DM process evaluation. More specifically, it contains descriptions of a comparison of the performances of all applied DM techniques, a comparison to previous work, and a discussion of the produced maps of vegetation properties.

The conclusion section, represented by the *conclusion to a publication about an investigation* instance, contains parts that are about the deployment process and determining further actions process, a sub-process of DM process evaluation. More specifically, it contains a summary of contributions, a description of the deployment of the produced maps, and a description of envisioned future work.

The considered example demonstrates that OntoDM-KDD has the expressivity to annotate the key concepts pertinent to typical DM investigations. OntoDM-KDD annotations would facilitate machine amenable recording of the information about how DM investigations have been carried out, enable accurate comparison of such investigations and reasoning e.g. about what DM methods work better for what applications. OntoDM-KDD is also an important resource to facilitate text mining of DM relevant literature.

[14] For simplicity/readability reasons, for all other parts Fig. 6 contains only the upper level processes.

8 Conclusion and Future Work

In this paper we proposed OntoDM-KDD, an ontology for representing the knowledge discovery based on the CRISP-DM process model. The OntoDM-KDD ontology was designed and implemented by following ontology best practices and design principles. It used an upper-level ontology BFO as a template, included formally defined relations, and reused classes from other ontologies for representing scientific investigations.

The ontology introduced a two-layered representation mechanism and provided a taxonomy of KD specific processes and actions. In addition, it provided a specification of inputs and outputs of the KD specific processes. Furthermore, the ontology introduced the data mining investigation entity as representational mechanism for describing and annotating data mining investigations in application domains (e.g., biology, forestry, etc). The OntoDM-KDD ontology has been applied for annotation of data mining investigations summarized in journal articles. In addition, the SWO ontology version 0.4 reused some of the OntoDM-KDD classes for representing data pre-processing and modeling processes.

In the context of representation of the complete knowledge discovery process, most of the current ontologies focus only on representing the modeling phase. Some of the ontologies, such as DMOP [13] and Expose [14], also provide entities that cover the data preparation phase of a KDD process, but do not provide ontological support for the complete KDD process. The strength of the OntoDM-KDD ontology is that it provides support for representation of the complete knowledge discovery process from application understanding to deployment.

In future developments of the OntoDM-KDD ontology, we plan to align the ontology to the BFO 2.0 top level ontology that is in final phases of preparations. Furthermore, we plan to apply the ontology for representation of data mining investigations in different application domains of data mining.

References

1. Yang, Q., Wu, X.: 10 challenging problems in data mining research. International Journal of Information Technologies and Decision Making 5(4), 597–604 (2006)
2. Kriegel, H.P., et al.: Future trends in data mining. Data Mining and Knowledge Discovery 15, 87–97 (2007)
3. Dietterich, T., Domingos, P., Getoor, L., Muggleton, S., Tadepalli, P.: Structured machine learning: the next ten years. Machine Learning 73 (2008)
4. Chapman, P., Kerber, R., Clinton, J., Khabaza, T., Reinartz, T., Wirth, R.: The CRISP-DM process model. Discussion Paper (1999)
5. King, R., et al.: The Automation of Science. Science 324(5923), 85–89 (2009)
6. Smith, B., et al.: The OBO foundry: coordinated evolution of ontologies to support biomedical data integration. Nat. Biotech. 25(11), 1251–1255 (2007)
7. Bernstein, A., Provost, F., Hill, S.: Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. IEEE Trans. on Knowl. and Data Eng. 17(4), 503–518 (2005)
8. Žáková, M., Kremen, P., Železný, F., Lavrač, N.: Automating knowledge discovery workflow composition through ontology-based planning. IEEE Transactions on Automation Science and Engineering 8(2), 253–264 (2010)

9. Diamantini, C., Potena, D.: Semantic annotation and services for KDD tools sharing and reuse. In: ICDMW 2008: Proceedings of the 2008 IEEE International Conference on Data Mining Workshops, pp. 761–770. IEEE Computer Society (2008)
10. Kietz, J., Serban, F., Bernstein, A., Fischer, S.: Towards cooperative planning of data mining workflows. In: Proceedings of Workshop on Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery, pp. 1–13 (2009)
11. Cannataro, M., Comito, C.: A data mining ontology for GRID programming. In: Proc. of Wshp. on Semantics in Peer-to-Peer and Grid Computing, pp. 113–134 (2003)
12. Brezany, P., Janciak, I., Tjoa, A.M.: Ontology-based construction of grid data mining workflows. In: Data Mining with Ontologies: Implementations, Findings and Frameworks, pp. 182–210. IGI Global (2007)
13. Hilario, M., et al.: A data mining ontology for algorithm selection and Meta-Mining. In: Proceedings of Workshop on Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery, pp. 76–88 (2009)
14. Vanschoren, J., Blockeel, H., Pfahringer, B., Holmes, G.: Experiment databases - a new way to share, organize and learn from experiments. Machine Learning 87(2), 127–158 (2012)
15. Panov, P., Džeroski, S., Soldatova, L.N.: OntoDM: An ontology of data mining. In: ICDMW 2008: Proceedings of the 2008 IEEE International Conference on Data Mining Workshops, pp. 752–760. IEEE Computer Society (2008)
16. Panov, P., Soldatova, L., Džeroski, S.: Representing entities in the OntoDM data mining ontology. In: Inductive Databases and Constraint-Based Data Mining, pp. 27–58. Springer, New York (2010)
17. Smith, B., et al.: Relations in biomedical ontologies. Genome Biology 6(5), R46 (2005)
18. Courtot, M., et al.: MIREOT: The minimum information to reference an external ontology term. Applied Ontology 6(1), 23–33 (2011)
19. Brinkman, R.R., et al.: Modeling biomedical experimental processes with obi. Journal of Biomedical Semantics 1(suppl. 1), S7 (2010)
20. Grüninger, M., Fox, M.: Methodology for the Design and Evaluation of Ontologies. In: IJCAI 1995, Workshop on Basic Ontological Issues in Knowledge Sharing (April 13, 1995)
21. Sirin, E., Parsia, B.: SPARQL-DL: Sparql query for OWL-DL. In: 3rd OWL Experiences and Directions Workshop (OWLED 2007) (2007)
22. Stojanova, D., Panov, P., Gjorgjioski, V., Kobler, A., Dzeroski, S.: Estimating vegetation height and canopy cover from remotely sensed data with machine learning. Ecological Informatics 5(4), 256–266 (2010)

A Wordification Approach to Relational Data Mining

Matic Perovšek[1,2], Anže Vavpetič[1,2], Bojan Cestnik[1,3], and Nada Lavrač[1,2,4]

[1] Jožef Stefan Institute, Ljubljana, Slovenia
[2] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
[3] Temida d.o.o., Ljubljana, Slovenia
[4] University of Nova Gorica, Nova Gorica, Slovenia
{matic.perovsek,anze.vavpetic,bojan.cestnik,nada.lavrac}@ijs.si

Abstract. This paper describes a propositionalization technique called *wordification*. Wordification is inspired by text mining and can be seen as a transformation of a relational database into a corpus of documents. Wordification aims at producing simple, easy to understand features, acting as words in the transformed Bag-Of-Words representation. As in other propositionalization methods, after the wordification step any propositional data mining algorithm can be applied. The most notable advantage of the presented technique is greater scalability: the propositionalization step is done in time linear to the number of attributes times the number of examples. The paper presents the wordification methodology, implemented in a cloud-based web data mining platform ClowdFlows, and describes the experiments in two real-life datasets together with a critical comparison to the RSD propositionalization approach.

Keywords: relational data mining, propositionalization, text mining, association rules, classification.

1 Introduction

Traditional data mining algorithms aim at finding models or patterns in a single data table (propositional patterns), whereas Inductive Logic Programming and Relational Data Mining is concerned with learning models or discovering interesting relational patterns from data stored in multiple tables. Most types of propositional models/patterns have corresponding relational models/patterns: relational classification rules, relational regression trees, relational association rules, etc.

For relational databases, where instances to be mined are clearly identifiable (so-called individual-centered relational databases, characterized by one-to-many relationships among data tables) there exist techniques for transforming such a database into a propositional or single-table format. After performing this transformation, called *propositionalization* [9, 5], traditional propositional learners can be used, such as decision tree learners or classification rule learners.

This paper introduces a propositionalization technique called *wordification*. As in other propositionalization methods, after the wordification step any propositional data mining algorithm can be applied. Unlike other propositionalization techniques [9, 5, 7, 17], which first construct complex relational features as constituents of subsequent propositional representation, wordification aims at generating simple features with greater scalability. Wordification is inspired by text mining techniques and can be seen as a transformation of a relational database into a corpus of documents, where each document is characterized by a set of properties describing the entries of a relational database. Wordification aims at producing simple, easy to understand features, acting as words in the transformed Bag-Of-Words representation. The feature construction step in wordification is very efficient, therefore it can scale well for large relational databases. In fact, the presented methodology transforms the database in time linear to the number of attributes times the number of examples. Furthermore, due to the simplicity of features, the generated features are easily interpretable by domain experts. Additionally, wordification uses Term Frequency-Inverse Document Frequency (TF-IDF) weighting [4] to capture the importance of a given feature (attribute value) of a relation in an aggregate manner.

As indicated in our early research on this topic [12], the wordification methodology suffers from loss of information, since the generated features do not capture existentially quantified variables connecting several relations. The research presented in this paper extends our previous research [12] by adding the ability of multiple feature word generation (partly overcoming the above mentioned loss of information problem), by presenting the methodology in greater detail, by adding more experimental results and by making the wordification methodology publicly available through its implementation as a reusable workflow in a cloud-based web data mining platform ClowdFlows[6], allowing for methodology reuse and experiment repeatability.

This paper presents the methodology and the results of experiments on two real-life relational databases: a collection of best and worst movies from the Internet Movie DataBase (IMDB) and a database of Slovenian traffic accidents. Furthermore, we provide a critical evaluation of the classification results (on the mutagenesis database) and running times of the wordification methodology, compared to another propositionalization technique RSD [17].

The paper is organized as follows. Section 2 presents the wordification methodology. The implementation of the methodology is described in Section 3. Section 4 presents the experimental results and Section 5 concludes the paper by presenting the plans for further work.

2 Wordification

This section presents the wordification methodology, and illustrates it with a simple example.

The transformation from a relational database to a text corpus is performed as illustrated in Figure 1. One text document represents one individual (i.e., one entry of the main table) of the initial relational database and the features (attribute

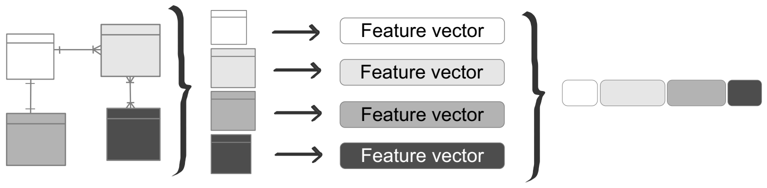


Fig. 1. The transformation from a relational database to a text corpus. One text document represents one individual (i.e., one entry of the main table) of the initial relational database and the features (attribute values) are taken as words of this document.

values) are taken as words of this document. One document is constructed simply as a list of attribute-value pairs: words (or features) constructed as a combination of a table name and the attribute name with its discrete (or discretized) value:

$$[table\ name]\_[attribute\ name]\_[value]. \tag{1}$$

Such constructs are called *word-items* or *witems* in the rest of the paper. Note that values of every non-discrete attribute need to be discretized beforehand in order to be able to represent them as word-items. For each individual, the word-items are first generated for the main table and then for each entry from the additional tables, and finally joined together according to the relational schema of the database.

In the described transformation a loss of information occurs as a consequence of building a "document" for each instance (an individual, i.e., a row in the main table) by concatenating all word-items from multiple instances (rows) of the connected tables to a single document. To overcome this loss, we extend the initial document construction step of our wordification methodology [12] by concatenating to the document also n-grams of word-items, constructed out of pairs and triplets of word-items. These concatenations of elementary word-items represent features that occur together in the instances/rows of the joined tables. In general, n-gram construction is performed by taking every combination of length n of word-items from the set of all word-items corresponding to the given individual, and concatenate them as follows:

$$[witem_1]\_[witem_2]\_\ ...\ \_[witem_n], \tag{2}$$

where each word-item is a combination of the table's name, name of the attribute and its discrete value. We concatenate the witems in a predetermined order.

Because we do not explicitly use existential variables in our new features (word-items), we instead rely on the Term Frequency-Inverse Document Frequency (TF-IDF) measure to implicitly capture the importance of a word-item for a given individual. In the context of text mining, TF-IDF value reflects how representative is a certain word (word-item) for a given document (individual). In the rest of this section, for simplicity, we refer to individuals as documents, and to word-items as witems or words. For a given witem w in document d from corpus D, the TF-IDF measure is defined as follows:

$$\mathrm{tfidf}(w,d) = \mathrm{tf}(w,d) \times \log \frac{|D|}{|\{d \in D : w \in d\}|}, \tag{3}$$

where $\mathrm{tf}(\cdot)$ represents the number of times w appears in document d. In other words, a witem with a high TF-IDF value will be considered important for the given individual provided that it is frequent within this document and not frequent in the entire corpus. In other words, the weight of a witem gives a strong indication of how relevant is the word-item for the given document. The TF-IDF weights can then be used either for filtering out word-items with low importance or using them directly by a propositional learner.

The technique is illustrated on a simplified version of the well-known East-West trains domain [10], where the input database consists of two tables shown in Table 1; we have one east-bound and one west-bound train, each with two cars with certain properties. The `Train` table is the main table and the trains are the individuals. We want to learn a classifier to determine the direction of an unseen train. For this purpose the direction attribute is not preprocessed and is only appended to the resulting feature vector (list of word-items).

Table 1. Example input for the standard East-West trains domain

Train

tid	direction
1	east
2	west

Car

cid	tid	shape	roof	wheels
1	1	rectangle	none	2
2	1	rectangle	peaked	3
3	2	rectangle	none	2
4	2	hexagon	flat	2

First, the corresponding two documents (one for each train) are generated, as shown in Figure 2. After this, the documents are transformed into a Bag-Of-Words representation by calculating the TF-IDF values for each word of each document using Equation 3, with the class attribute column appended to the transformed Bag-Of-Words table. For simplicity, only unigrams are shown in this example.

1 [car_shape_rectangle, car_roof_none, car_wheels_2, car_shape_rectangle, car_roof_peaked, car_wheels_3], class: east
2 [car_shape_rectangle, car_roof_none, car_wheels_2, car_shape_hexagon, car_roof_flat, car_wheels_2], class: west

Fig. 2. The database from Table 1 in document representation

3 Implementation

We implemented the wordification methodology in a cloud-based web platform, named ClowdFlows [6]. ClowdFlows, in contrast to other data mining platforms, runs in a browser, which in terms of crowdsourcing, provides researchers with an easy way to expose and share their data, workflows and results. ClowdFlows allows researchers to seamlessly integrate different implementations of algorithms, tools and web services into a coherent workflow that can be executed in a cloud-based application. Existing data mining workflow components are separated into modules (e.g., Orange [2] module, Weka [16] module, NLP module, ILP module, etc.), which are the sets of components designed for diverse tasks.

The existing ClowdFlows ILP module includes components, such as the popular ILP system Aleph [14], as well as RSD [17] and SDM [15]. Aleph is an ILP toolkit by itself with a wide range of uses: from decision tree learning to feature generation and first-order rule induction. Relational Subgroup Discovery (RSD) algorithm implements a propositionalization approach. It starts with a typical relational ILP domain and converts it into a single-table representation; this is done by generating a set of first-order features which become attributes of the propositionalized training examples. Although RSD comes with its own implementation of the CN2-SD [8] subgroup discovery algorithm, the resulting table can serve an input into any propositional machine learning or data mining algorithm.

We have extended the ClowdFlows ILP module with the implementation of the *wordification* component, which together with the existing components from different modules of the ClowdFlows platform forms the entire workflow of the wordification methodology, as can be seen in Figure 3.

3.1 Workflow

This section describes the main components of the wordification workflow, which is shown on Figure 3. The implementation allows the users to provide as input a relational database by connecting to a MySQL database server. First, the user is required to select the target table from the initial relational database, which will later represent the main table in the Wordification component of the workflow. Second, the user is able to discretize each table using one of the various discretization techniques provided. These discretized tables are used in the last part of the workflow—the Wordification widget—where the transformation from the relational tables to a corpus of documents composed of word-items is performed. In the following paragraphs every component of the presented workflow is described in more detail.

MySQL Connect. Since relational data is often stored in SQL databases, we use the MySQL package to access the training data by connecting to a MySQL database server. The MySQL Connect widget is used for entering information required to connect to a database (e.g., user credentials, database address, database name, etc.) in order to retrieve the training data from a MySQL

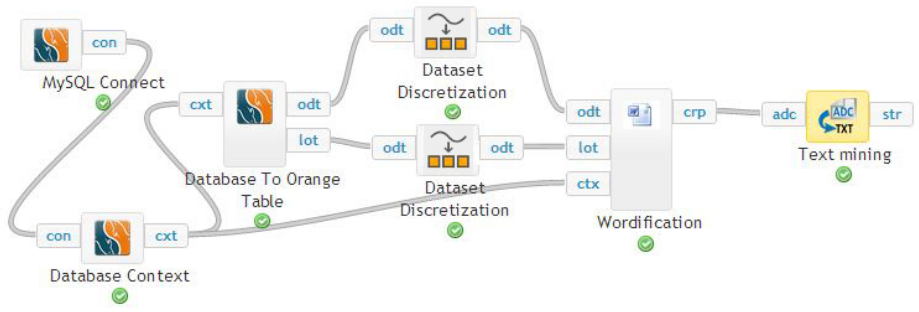


Fig. 3. ClowdFlows implementation of the wordification methodology

database server and automatically construct the background knowledge and the training examples.

Database Context. This widget enables a selection of tables and columns that will be used in the next steps of the methodology. The information is carried to the connected widgets through the so-called database context objects. These objects also contain the detected table relationships. In case that the input relational database does not have preset primary and foreign keys between the tables, the user is given an option for a simple table connection search through the names of the attributes.

Database to Orange Table. The task of this widget is to transform the given database context to an Orange dataset, which is a required step for next components in the wordification workflow.

Dataset Discretization. This sole task of this widget is to convert continuous attributes to categorical. In other words, it discretizes the continuous attributes. The Dataset Discretization widget supports three discretization methods: using equal-width intervals, using equal-frequency intervals, and class-aware discretization of Fayyad and Irani [3] that uses MDL and entropy to find the best cut-off points. Dataset Discretization widget can take as an input a single data set (Orange dataset) or a list of multiple datasets. In the latter case, discretization of all continuous attributes of every dataset is performed.

Wordification. The wordification widget transforms the relational database to a corpus of documents for the main table. As an input it takes three arguments: the target (main) table, a list of additional tables and a database context, which contains the relations between the tables. The wordification widget first indexes the examples of every table by their primary and foreign keys' values. This step is required to improve the speed of data retrieval operations, when searching for connecting instances from different (connected) tables in the word-item concatenation step. Next, recursive document construction for every individual is

Table 2. Document properties after applying the wordification methodology

Domain	Individual	#examples	#words	#words after filtering
IMDB	movie	166	7,453	3,234
Accidents	accident	102,759	186	79

performed. The algorithm starts on every example of the main table: it creates word-items for its features, which is followed by concatenations of the word-items and results of the recursive search through the connecting tables. When searching along the tree of the connected tables the algorithm stores the results of subtree word-item concatenations for every instance. Consequently, the algorithm iterates over every subtree only once.

Text analysis. After the wordification step the user can perform various types of text analysis, depending on the task at hand (e.g,. text categorization, text clustering, etc.).

4 Experiments

In Section 4.1 we present our association rule learning experiments on two real-life relational databases: a collection of best and worst movies from the Internet Movie DataBase (IMDB) and a database of Slovenian traffic accidents. Additionally, we performed classification experiments on a more complex and better researched relational database in the ILP comunity: the mutagenesis dataset. We provide a critical evaluation of the results and running times of the wordification methodology, compared to another propositionalization technique RSD in Section 4.2.

4.1 Association Rule Learning

In the following paragraphs we will present the results of the association rule learning after applying the wordification methodology. We performed association rule learning in combination with our wordification approach on two real-life datasets: the best and worst ranked IMDB movies database and the Slovenian traffic accidents database Table 2 and Table 3 list the characteristics of both databases.

The preprocessing procedure was performed on both databases as follows. First, the wordification step was applied as described in Section 2. Next, irrelevant features (which have the same value across all examples) were removed, resulting in the reduction of the features to less than half of the original (see Table 2). In order to prepare the data for association rule mining, we also binarized the data: after experimenting with different TF-IDF thresholds, features with a corresponding TF-IDF weight were assigned the value *true* and *false* otherwise.

Table 3. Table properties of the experimental data

IMDB	#rows	#attributes
movies	166	4
roles	7,738	2
actors	7,118	4
movie_genres	408	2
movie_directors	180	2
directors	130	3
director_genres	243	3

Accidents	#rows	#attributes
accident	102,756	10
person	201,534	10

IMDB Database. The complete IMDB database is publicly available in the SQL format[1]. This database contains tables of movies, actors, movie genres, directors, director genres.

The evaluation database used in our experiments consists only of the movies whose titles and years of production exist on IMDB's top 250 and bottom 100 chart[2]. The database therefore consisted of 166 movies, along with all of their actors, genres and directors. Movies present in the IMDB's top 250 chart were added an additional label *goodMovie*, while those in the bottom 100 were marked as *badMovie*. Additionally, attribute age was discretized; a movie was marked as *old* if it was made before 1950, *fairlyNew* if it was produced between 1950 and 2000 and *new* otherwise.

```
goodMovie ← director_genre_drama, movie_genre_thriller,
            director_name_AlfredHitchcock. (Support: 5.38% Confidence: 100.00%)

movie_genre_drama ← goodMovie, actor_name_RobertDeNiro.
(Support: 3.59% Confidence: 100.00%)

director_name_AlfredHitchcock ← actor_name_AlfredHitchcock.
(Support: 4.79% Confidence: 100.00%)

director_name_StevenSpielberg ← goodMovie, movie_genre_adventure,
(Support: 1.79% Confidence: 100.00%)    actor_name_TedGrossman.
```

Fig. 4. Examples of interesting association rules discovered in the IMDB database

After preprocessing the dataset using the wordification methodology, we performed association rule learning. Frequent item sets were generated using RapidMiner's [11] FP-growth implementation. Next, association rules for the resulting frequent item sets were produced. Among all the discovered rules, several interesting rules were found. Figure 4 presents some of the interesting rules selected by the experts. The first rule states that if the movie's genre is thriller and is directed by Alfred Hitchcock, who is also known for drama movies, then the

[1] `http://www.webstepbook.com/supplements/databases/imdb.sql`

[2] As of July 2, 2012.

movie is considered as good. The second rule we have selected concludes that if the movie is good and Robert De Niro acts in it, than it must be a drama. The third interesting rule we present, shows that Alfred Hitchcock acts only in the movies he also directs. The last rule concludes, that if Ted Grossman acts in a good adventure movie, then the director is Steven Spielberg. Note that Ted Grossman usually plays the role of a stunt coordinator or performer.

Traffic Accident Database. The second dataset consists of all accidents that happened in Slovenia's capital city Ljubljana between the years 1995 and 2005. The data is publicly accessible from the national police department website[3]. The database contains the information about accidents along with all the accident's participants.

```
noInjuries ← accident_trafficDensity_rare,
             accident_location_parkingLot. (Support: 0.73% Confidence: 97.66%)

person_gender_male ← person_vehicleType_motorcycle.
(Support: 0.11% Confidence: 99.12%)
```

Fig. 5. Examples of interesting association rules discovered in the accidents database

The data already contained discretized attributes, so further discretization was not needed. Similarly to the IMDB databse, preprocessing using wordification methodology, FP-growth itemset mining and association rule mining were performed. Figure 4.1 presents some of the interesting rules found in the Slovenian traffic accidents dataset.

The first rule indicates that if the traffic is rare and the accident happened in a parking lot, then no injuries occurred. The second rule implies that whenever a motorcycle is involved in an accident, a male person is involved.

4.2 Classifier Evaluation on Mutagenesis Data

This section presents a critical evaluation of the wordification methodology. First we provide a description of the well researched relational database in the ILP comunity: the mutagenesis dataset. Next, we describe the experiments performed on this real-world dataset and provide a comparison of the wordification and RSD propositionalization techniques.

Mutagenesis Dataset. The purpose of the mutagenesis dataset is to predict the mutagenicity of a set of 230 aromatic and heteroaromatic nitro compounds [1]. Predicting the mutagenesis is an important task as it is much relevant to the prediction of carcinogenesis. The compounds from the data are known to be more structurally heterogeneous than in any other ILP dataset of chemical

[3] `http://www.policija.si/index.php/statistika/prometna-varnost`

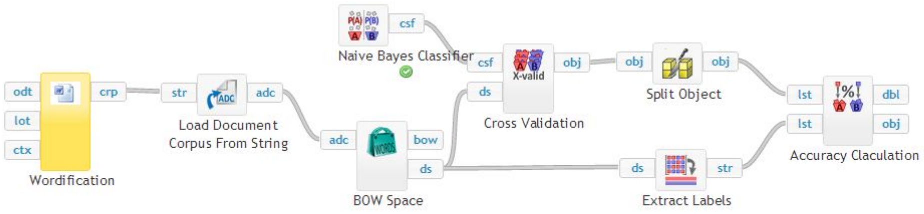


Fig. 6. Classification of text data using the Latino module in ClowdFlows

structures. The database contains 230 compounds of which 138 have positive levels of mutagenicity and are labelled as "active". Others have class value "inactive" and are considered to be negative examples.

We took the datasets of the original Debnath paper [1], where the data was split into two subsets of data: a 188 compound dataset and a smaller dataset with 42 compounds. In the following paragraph we describe the required steps compound classification on the two datasets.

As described in Section 3, the first step in our methodologiy is the preprocessing step of the original tables of the relational database. After experimenting with different discretization settings equi-distance discretization with 3 intervals of values was used to discretize the continuous attributes of the mutagenesis dataset. The wordification propositionalization step was tested with three different parameter settings: using elementary witems, 2-grams of witems and 3-grams of witems. After applying the wordification algorithm, we used ClowdFlow's module Latino to categorize the documents in order to perform text classification (categorization). Figure 6 shows the constructed classification workflow after applying the wordification propositionalization. First, the BOW Space widget was used to transform the documents into a Bag-Of-Words representation by calculating the TF-IDF values for each word of each document using Equation 3. The class attribute column was appended to the transformed Bag-Of-Words table. The minimum word frequency was set to 2–words that occurred only once in all documents were discarded. TF-IDF values were also calculated for bigrams of words. Stop word removal was not required, as all words were artificially created.

As shown in Figure 7, concurrent evaluation of the RSD propositionalization step was performed. The RSD widget contstructed features with a specified maximum length of a feature body of 8. None of the constructed feature were discarded as the minimum example coverage of the algortihm was set to 1.

To perform the classification on the constructed propositional datasets we used the standard Naive Bayes classifier. Classification accuracies of the algorithm for the two datasets (the 188 example and 42 example mutagenesis datasets) were estimated with the leave-one-out validation.

The results (Table 4) of the experiments on the mutagenesis datasets show that wordification methodology achieved scores comparable to a more complex propositionalization technique RSD, while the run-time required for transforming the database into its propositional form was much lower. Using 2-grams

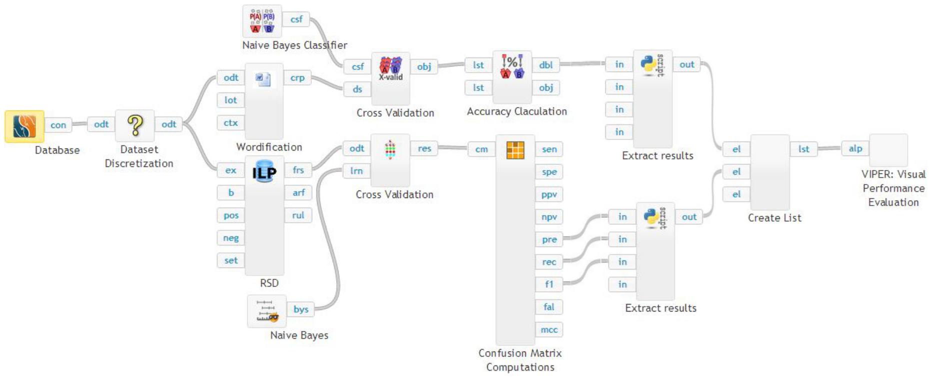


Fig. 7. The evaluation workflow

and 3-gram witems aimed at improving the classification accuracies, though resulting also in longer run-times of the propositionalization step. Similar results were achieved on the smaller (42 example) mutagenesis dataset, though with a smaller run-time difference of the propositionalization step when comparing the two methodologies. We could therefore argue that the added value of the wordification methodology would show more drastically on even larger datasets, where other propositionalization approaches, because of their higher time complexity, might ran into difficulties when trying to obtain the results in sufficient time.

As shown on Figure 7, the results of Confusion Matrix Computations widgets (precision, recall, and F-score) are extracted using the Extract Results widget and stored on the outputs of these widgets in forms of lists. These results are concatenated in the Create list widget. Next, the VIPER (Visual Performance Evaluation) [13] widget displays the results as a point in the two dimensional precision-recall space for the positive examples.

Figure 8 presents the VIPER performance visualization, evaluating the standard Naive Bayes after applying wordification and RSD as propositionalizaton techniques. The results are presented in the so-called precision-recall space, where each point represents an evaluation of an algorithm. Points closer to the upper-right corner have higher precision and recall values. F-measure values are

Table 4. Classifier evaluation on different databases

Database	Algorithm	CA	Precision	Recall	F-measure	Runtime
Mutagenesis 188	Wordification (3-gram witems)	0.723	0.708	1.000	0.829	7.8
Mutagenesis 188	Wordification (2-gram witems)	0.723	0.708	1.000	0.829	7.7
Mutagenesis 188	Wordification (1-gram witems)	0.718	0.702	1.000	0.825	7.4
Mutagenesis 188	RSD	0.745	0.798	0.824	0.811	21.3
Mutagenesis 42	Wordification (3-gram witems)	0.881	1.000	0.615	0.762	5.1
Mutagenesis 42	Wordification (2-gram witems)	0.881	1.000	0.615	0.762	5.1
Mutagenesis 42	Wordification (1-gram witems)	0.881	1.000	0.615	0.762	4.9
Mutagenesis 42	RSD	0.881	0.833	0.769	0.800	5.7

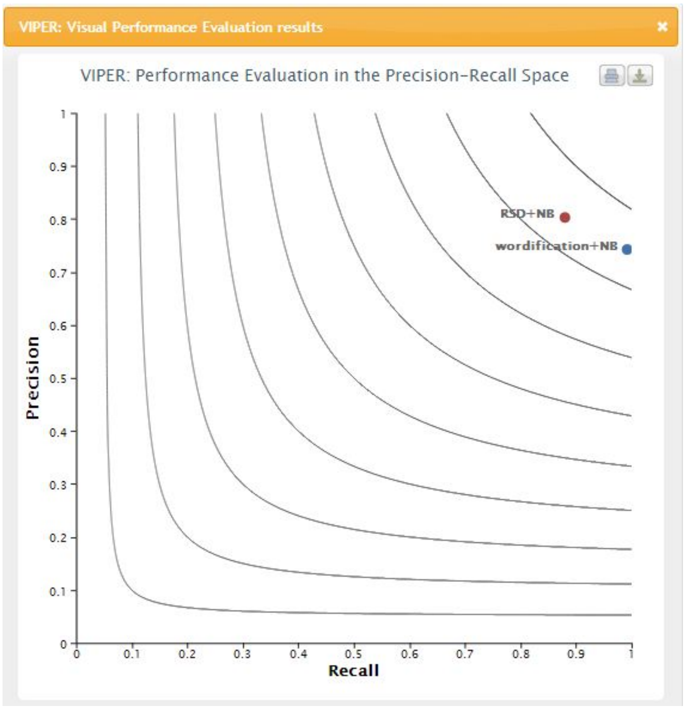


Fig. 8. The VIPER visualization showing evaluations of the standard Naive Bayes algorithm after applying wordification and RSD as propositionalizaton techniques on the 188 example mutagenesis dataset

presented as isolines (contour lines) in the precision-recall space, which allows a simple comparison of algorithm performances. From the results shown on figure 8 we can conclude that there is not much difference between RSD and wordification as they are located on the same isoline. Using the wordification methodology a higher percentage of positive examples were retrieved (higher recall score) compared to the approach using RSD, while the latter correctly classified a slightly higher percentage of positive examples (higher precision score).

5 Conclusion

This paper presents a novel propositionalization technique called wordification which aims at producing simple and easy to understand features. This methodology is inspired by text mining and can be seen as a transformation of a relational database into a corpus of documents. As is typical for propositionalization methods, after the wordification step any propositional data mining algorithm can be applied.

We have presented initial results on two real-life databases. Namely, best and worst movies from the IMDB database and a database of Slovenian traffic accidents. Using association rule learning on the IMDB data we found interesting and easily interpretable patterns using our methodology. The results of the experiments on the mutagenesis datasets confirm the most notable advantage of the presented technique–its greater scalability, as the propositionalization step is done in time linear to the number of attributes times the number of examples. The wordification methodology achieves comparable results to more complex propositionalization techniques with less time required for the propositionalization step. We expect that the value of the wordification methodology will prove to be even more favorable on even larger datasets, where other propositionalization approaches may show their limitations when trying to get results in a limited time constraints.

In future work we plan to apply the methodology on larger databases to further explore its advantages and potential limitations, and will experimentally compare the proposed methodology with other relational data mining techniques, such as RELF [7] and Aleph [14].

References

[1] Debnath, A.K., Lopez de Compadre, R.L., Debnath, G., Shusterman, A.J., Hansch, C.: Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. Journal of Medicinal Chemistry 34(2), 786–797 (1991)

[2] Demšar, J., Zupan, B., Leban, G., Curk, T.: Orange: From Experimental Machine Learning to Interactive Data Mining. Springer (2004)

[3] Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, pp. 1022–1027 (1993)

[4] Jones, K.S.: A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation 28, 11–21 (1972)

[5] Kramer, S., Pfahringer, B., Helma, C.: Stochastic propositionalization of nondeterminate background knowledge. In: Page, D.L. (ed.) ILP 1998. LNCS, vol. 1446, pp. 80–94. Springer, Heidelberg (1998)

[6] Kranjc, J., Podpečan, V., Lavrač, N.: Clowdflows: a cloud based scientific workflow platform. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part II. LNCS, vol. 7524, pp. 816–819. Springer, Heidelberg (2012)

[7] Kuželka, O., Železný, F.: Block-wise construction of tree-like relational features with monotone reducibility and redundancy. Machine Learning 83(2), 163–192 (2011)

[8] Lavrač, N., Kavšek, B., Flach, P., Todorovski, L.: Subgroup discovery with cn2-sd. The Journal of Machine Learning Research 5, 153–188 (2004)

[9] Lavrač, N., Džeroski, S., Grobelnik, M.: Learning nonrecursive definitions of relations with LINUS. In: Kodratoff, Y. (ed.) EWSL 1991. LNCS, vol. 482, pp. 265–281. Springer, Heidelberg (1991)

[10] Michie, D., Muggleton, S., Page, D., Srinivasan, A.: To the international computing community: A new east-west challenge. Technical report, Oxford University Computing laboratory, Oxford, UK (1994)

[11] Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: YALE: Rapid prototyping for complex data mining tasks. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD 2006), Philadelphia, PA, USA, August 20-23, pp. 935–940. ACM Press, NY (2006)
[12] Perovšek, M., Vavpetič, A., Lavrač, N.: A wordification approach to relational data mining: Early results. In: Late Breaking Papers of the 22nd International Conference on Inductive Logic Programming, pp. 56–61 (2012)
[13] Sluban, B., Gamberger, D., Lavrač, N.: Ensemble-based noise detection: noise ranking and visual performance evaluation. In: Data Mining and Knowledge Discovery 2013, pp. 1–39 (2013)
[14] Srinivasan, A.: Aleph manual (March 2007), `http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/`
[15] Vavpetič, A., Lavrač, N.: Semantic subgroup discovery systems and workflows in the sdm-toolkit. The Computer Journal 56(3), 304–320 (2013)
[16] Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques. Morgan Kaufmann (2011)
[17] Železný, F., Lavrač, N.: Propositionalization-based relational subgroup discovery with RSD. Machine Learning 62, 33–63 (2006)

Multi-interval Discretization of Continuous Attributes for Label Ranking

Cláudio Rebelo de Sá[1,2], Carlos Soares[1],
Arno Knobbe[2], Paulo Azevedo[3], and Alípio Mário Jorge[4]

[1] INESCTEC, Porto, Portugal
[2] LIACS, Universiteit Leiden, Netherlands
[3] CCTC, Departamento de Informática, Universidade do Minho
[4] DM - Faculdade de Ciencias, Universidade do Porto
{claudio,knobbe}@liacs.nl, csoares@fe.up.pt, pja@uminho.pt,
amjorge@fc.up.pt

Abstract. Label Ranking (LR) problems, such as predicting rankings of financial analysts, are becoming increasingly important in data mining. While there has been a significant amount of work on the development of learning algorithms for LR in recent years, pre-processing methods for LR are still very scarce. However, some methods, like Naive Bayes for LR and APRIORI-LR, cannot deal with real-valued data directly. As a make-shift solution, one could consider conventional discretization methods used in classification, by simply treating each unique ranking as a separate class. In this paper, we show that such an approach has several disadvantages. As an alternative, we propose an adaptation of an existing method, MDLP, specifically for LR problems. We illustrate the advantages of the new method using synthetic data. Additionally, we present results obtained on several benchmark datasets. The results clearly indicate that the discretization is performing as expected and in some cases improves the results of the learning algorithms.

1 Introduction

A reasonable number of learning algorithms has been created or adapted for LR in recent years [15,11,7,4,5]. LR studies the problem of learning a mapping from instances to rankings over a finite number of predefined labels. It can be considered as a variant of the conventional classification problem [3]. However, in contrast to a classification setting, where the objective is to assign examples to a specific class, in LR we are interested in assigning a complete preference order of the labels to every example. An additional difference is that the true (possibly partial) ranking of the labels is available for the training examples.

Discretization, from a general point of view, is the process of partitioning a given interval into a set of discrete sub-intervals. It is usually used to split continuous intervals into two or more sub-intervals which can be treated as nominal values. This pre-processing technique enables the application to numerical data of learning methods that are otherwise unable to process them directly (like

Bayesian Networks and Rule Learning methods). In theory, a good discretization should have a good balance between the loss of information and the number of partitions [13].

Discretization methods come in two flavors, depending on whether they do, or do not involve target information. These are usually referred to as *supervised* and *unsupervised*, respectively. Previous research found that the supervised methods produce more accurate discretization than unsupervised methods [8]. To the best of our knowledge, there are no supervised discretization methods for LR. Hence, our proposal for ranking-sensitive discretization is a useful contribution for the LR community.

We propose an adaptation of a well-known supervised discretization method, the Minimum Description Length Principle (MDLP) [10], for LR. The method uses an entropy-like measure for a set of rankings based on a similarity measure for rankings. Despite this basic heuristic approach, the results observed show that the method is behaves as expected in an LR setting.

The paper is organized as follows: Section 2 introduces the LR problem and the task of association rule mining. Section 3 introduces discretization and Section 4 describes the method proposed here. Section 5 presents the experimental setup and discusses the results. Finally, Section 6 concludes this paper.

2 Label Ranking

The LR task is similar to classification. In classification, given an instance x from the instance space $\mathbb{X}$, the goal is to predict the label (or class) λ to which x belongs, from a pre-defined set $\mathcal{L} = \{\lambda_1, \ldots, \lambda_n\}$. In LR the goal is to predict the ranking of the labels in $\mathcal{L}$ that are associated with x. We assume that the ranking is a total order over $\mathcal{L}$ defined on the permutation space Ω. A total order can be seen as a permutation π of the set $\{1, \ldots, n\}$, such that $\pi(a)$ is the position of λ_a in π.[1]

As in classification, we do not assume the existence of a deterministic $\mathbb{X} \rightarrow \Omega$ mapping. Instead, every instance is associated with a *probability distribution* over Ω. This means that, for each $x \in \mathbb{X}$, there exists a probability distribution $P(\cdot|x)$ such that, for every $\pi \in \Omega$, $P(\pi|x)$ is the probability that π is the ranking associated with x. The goal in LR is to learn the mapping $\mathbb{X} \rightarrow \Omega$. The training data is a set of instances $T = \{\langle x_i, \pi_i \rangle\}, i = 1, \ldots, n$, where x_i are the independent variables describing instance i and π_i is the corresponding target ranking.

Given an instance x with label ranking π, and the ranking $\hat{\pi}$ predicted by an LR model, we need to evaluate the accuracy of the prediction. For that, we need a loss function on Ω. One such function is the number of discordant label pairs,

$$D(\pi, \hat{\pi}) = \#\{(i,j)|\pi(i) > \pi(j) \wedge \hat{\pi}(i) < \hat{\pi}(j)\}$$

which, if normalized to the interval $[-1, 1]$, is equivalent to Kendall's τ coefficient [12], which is a correlation measure where $D(\pi, \pi) = 1$ and $D(\pi, \pi^{-1}) = -1$ (here, π^{-1} denotes the inverse order of π).

[1] This assumption may be relaxed [3].

The accuracy of a model can be estimated by averaging this function over a set of examples. This measure has been used for evaluation in recent LR studies [3] and, thus, we will use it here as well. However, other correlation measures, like Spearman's rank correlation coefficient [16], can be used equally well, were one so inclined.

Given the similarities between LR and classification, one could consider workarounds that treat the label ranking problem essentially as a classification problem.

Let us define a basic pre-processing method, which replaces the rankings with classes, as Ranking As Class (RAC).

$$\forall \pi_i \in \Omega, \pi \rightarrow \lambda_i$$

This method has a number of disadvantages, as discussed in the next section, but it allows the use of all pre-processing and prediction methods for classification in LR problems. However, as we show in this work, this approach is neither the most effective nor the most accurate.

2.1 Association Rules for Label Ranking

Label Ranking Association Rules (LRAR) [7] are a straightforward adaptation of class Association Rules (CAR):

$$A \rightarrow \pi$$

where $A \subseteq desc(\mathbb{X})$ and $\pi \in \Omega$. Similar to how predictions are made in CBA (Classification Based on Associations) [14], when an example matches the rule $A \rightarrow \pi$, the predicted ranking is π.

If the RAC method is used, the number of classes can be extremely large, up to a maximum of $k!$, where k is the size of the set of labels, $\mathcal{L}$. This means that the amount of data required to learn a reasonable mapping $\mathbb{X} \rightarrow \Omega$ is too big.

Secondly, this approach does not take into account the differences in nature between label rankings and classes. In classification, two examples either have the same class or not, whereas in LR some rankings are more similar than others, as they only differ in one or two swaps or labels. In this regard, LR is more similar to regression than to classification. This property can be used in the induction of prediction models. In regression, a large number of observations with a given target value, say 5.3, increases the probability of observing similar values, say 5.4 or 5.2, but not so much for very different values, say -3.1 or 100.2. A similar reasoning was done for LR in [7]. Let us consider the case of a data set in which ranking $\pi_a = \{A, B, C, D, E\}$ occurs in 1% of the examples. Treating rankings as classes would mean that $P(\pi_a) = 0.01$. Let us further consider that the rankings $\pi_b = \{A, B, C, E, D\}, \pi_c = \{B, A, C, D, E\}$ and $\pi_d = \{A, C, B, D, E\}$ occur in 50% of the examples. Taking into account the stochastic nature of these rankings [3], $P(\pi_a) = 0.01$ seems to underestimate the probability of observing π_a. In other words it is expected that the observation of π_b, π_c and π_d increases the probability of observing π_a and vice-versa, because they are similar to each other.

This affects even rankings which are not observed in the available data. For example, even though $\pi_e = \{A, B, D, C, E\}$ is not present in the data set it would not be entirely unexpected to see it in future data.

Similarity-Based Support and Confidence. Given a measure of similarity between rankings $s(\pi_a, \pi_b)$, the *support* of the rule $A \rightarrow \pi$ is defined as follows:

$$sup_{lr}(A \rightarrow \pi) = \frac{\sum\limits_{i:A \subseteq desc(x_i)} s(\pi_i, \pi)}{n}$$

This is, essentially, assigning a weight to each target ranking in the training, π_i, data that represents its contribution to the probability that π may be observed. Some instances $x_i \in \mathbb{X}$ give full contribution to the support count (i.e., 1), while others may give partial or even a null contribution.

Any function that measures the similarity between two rankings or permutations can be used, such as Kendall's τ or Spearman's ρ. The function used here is of the form:

$$s(\pi_a, \pi_b) = \begin{cases} s'(\pi_a, \pi_b) & \text{if } s'(\pi_a, \pi_b) \geq \theta_{sup} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where s' is a similarity function. This general form assumes that below a given threshold, θ_{sup}, it is not useful to discriminate between different similarity values, as they are so different from π_a. This means that, the support *sup* of $\langle A, \pi_a \rangle$ will have contributions from all the *ruleitems* of the form $\langle A, \pi_b \rangle$, for all π_b where $s'(\pi_a, \pi_b) > \theta_{sup}$.

The *confidence* of a rule $A \rightarrow \pi$ is obtained simply by replacing the measure of support with the new one.

$$conf_{lr}\left(A \rightarrow \pi\right) = \frac{sup_{lr}\left(A \rightarrow \pi\right)}{sup\left(A\right)}$$

Given that the loss function that we aim to minimize is known beforehand, it makes sense to use it to measure the similarity between rankings. Therefore, we use Kendall's τ. In this case, we think that $\theta_{sup} = 0$ would be a reasonable value, given that it separates the negative from the positive contributions. Table 1 shows an example of a label ranking dataset represented following this approach.

To present a more clear interpretation, the example given in Table 1, the instance

$$\left(\{A1 = L, A2 = XL, A3 = S\}\right)\left(TID = 1\right)$$

contributes 1 to the support count of the ruleitem:

$$\langle \{A1 = L, A2 = XL, A3 = S\}, \pi_3 \rangle$$

Table 1. An example of a label ranking dataset to be processed by the APRIORI-LR algorithm

				π_1	π_2	π_3
TID	A1	A2	A3	$(1,3,2)$	$(2,1,3)$	$(2,3,1)$
1	L	XL	S	0.33	0.00	1.00
2	XXL	XS	S	0.00	1.00	0.00
3	L	XL	XS	1.00	0.00	0.33

The same instance will also give a small contribution of 0.33 to the support count of the ruleitem

$$\langle\{A1 = L, A2 = XL, A3 = S\}, \pi_1\rangle$$

given their similarity. On the other hand, no contribution is given to the count used for the support of ruleitem

$$\langle\{A1 = L, A2 = XL, A3 = S\}, \pi_2\rangle$$

which makes sense as they are clearly different.

3 Discretization

Several Data Mining (DM) algorithms can improve their performance by using discretized versions of continuous-valued attributes [9]. Given that a large number of algorithms, like the Naive Bayes classifier, cannot work without discretized data [13] and the majority of real datasets have continuous variables, a good discretization method can be very relevant for the accuracy of the models. Discretization methods deal with continuous variables by partitioning them into intervals or ranges. Then, each of these intervals can be interpreted as a nominal value by DM algorithms.

The main issue in discretization is the choice of the intervals because a continuous variable can be discretized in an infinite number of ways. An ideal discretization method finds a reasonable number[2] of cut points that split the data into meaningful intervals. For classification datasets, a meaningful interval should be coherent with the class distribution along the variable.

Discretization approaches can be divided into two groups:

Supervised vs Unsupervised. When dealing with classification datasets the discretization methods can use the values of the target variable or not. These are referred to as *supervised* and *unsupervised* respectively. The *unsupervised* methods ignore the classes of the objects and divide the interval into a user-defined number of bins. *Supervised* methods take into account the distribution of the class labels in the discretization process. Previous research states that the supervised methods tend to produce better discretizations than unsupervised methods [8].

[2] An extreme discretization approach would create one nominal value for each continuous value but this is naturally not a reasonable approach.

Top-down vs Bottom-up. Discretization methods with a Top-down or Bottom-up approach start by sorting the dataset with respect to the variable which will be discretized. In the Top-down approach, the method starts with an interval containing all points. Then, it recursively splits the intervals into sub-intervals, until a stopping criteria is reached.

In the Bottom-up approach, the method starts with the maximum number of intervals (i.e., one for each value) and then iteratively merges them recursively until a stopping criteria is satisfied.

3.1 Entropy Based Methods

Several methods, such as [6,10], perform discretization by optimizing entropy. In classification, class entropy is a measure of uncertainty in a finite interval of classes and it can be used in the search of candidate partitions. A good partition is such that it minimizes the overall entropy in its subsets. Likewise, in discretization, a good partition of the continuous variable minimizes the class entropy in the subsets of examples it creates. In [10] it was shown that optimal cut points must be between instances of distinct classes. In practical terms, for all possible partitions the class information entropy is calculated and compared with the entropy without partitions. This can be done recursively until some stopping criterion is satisfied. The stopping criteria can be defined by a user or by a heuristic method like MDLP.

4 Discretization for Label Ranking

A supervised discretization method for LR should take into account the properties of rankings as target variables. In this work, we propose an adaptation of the Shannon entropy for rankings. This entropy will be used in conjuction with MDLP as stopping criterion, the same way it is used for classification. First we describe our adaptation of the entropy for rankings and then we show how to integrate it with MDLP.

The entropy of classes presented in [10], which derives from the Shannon entropy, is defined as:

$$Ent\,(S) = -\sum_{i=1}^{k} P\,(C_i, S)\,log\,(P\,(C_i, S)) \tag{2}$$

where $P\,(C_i, S)$ stands for the proportion of examples with class C_i in a subset S and k is the total number of classes in S.

$$P\,(C_i, S) = \frac{\#C_i}{N}$$

N is the number of instances in the subset S.

As shown in equation 2 the Shannon entropy of a set of classes depends on the relative proportion of each class.

4.1 Entropy of Rankings

In this section, we explain how to adapt the entropy of classes used in [10] for LR. We start by motivating our approach with a discussion of the use in LR of the concept of entropy from classification. We then show in detail our heuristic adaptation of entropy for rankings.

To better motivate and explain our approach, we introduce a very simple synthetic dataset, D_{ex}, presented in Table 2. In this test dataset we have eight distinct rankings in the target column π. Even though they are all distinct, the first five are very similar (the label ranks are mostly ascending), but very different from the last three (mostly descending ranks). Without any further considerations, it is natural to assume that an optimal split point for D_{ex} should lie between values 0.5 and 0.6 (instances 5 and 6).

Table 2. Example dataset D_{ex}: Small artificial dataset with some noise in the rankings

TID	Att	π	λ
1	0.1	(1,2,4,3,5)	a
2	0.2	(1,2,3,4,5)	b
3	0.3	(2,1,3,4,5)	c
4	0.4	(1,3,2,4,5)	d
5	0.5	(1,2,3,5,4)	e
6	0.6	(5,4,3,1,2)	f
7	0.7	(4,5,3,2,1)	g
8	0.8	(5,3,4,2,1)	h

In the RAC approach, the rankings are transformed into eight distinct classes as shown in column λ. As the table shows, the natural split point identified earlier is completely undetectable in column λ.

As shown in equation 2, the entropy of a set of classes depends on the relative proportion of a class. If we measure the ranking proportion the same way, we get:

$$P(\pi_i, S) = 1/8, \forall \pi_i \in D_{ex}$$

We adapt this concept using the same ranking distance-based approach used to adapt the support for LRAR in APRIORI-LR [7] (equation 3). In fact, a similar line of reasoning as the one in Section 2.1 can be followed here. The uncertainty associated with a certain ranking decreases in the presence of similar – although not equal – rankings. Furthermore, this decrease is proportional to that distance.

$$P_\pi(\pi_i, S) = \frac{\sum_{j=1}^{N} s(\pi_i, \pi_j)}{\sum_{i=1}^{K} \sum_{j=1}^{N} s(\pi_i, \pi_j)} \qquad (3)$$

Where K is the number of distinct rankings in S.

As in [7], we use Kendall τ and the negative correlations are ignored (section 2.1). Note that a parallel can also be established with the frequentist view used in entropy. Since Kendall τ is computed from the proportion of concordant pairs of labels, this can be seen as the proportion of concordant pairwise comparisons.

However, this approach alone is not enough to give a fair measure for the entropy of rankings. The entropy of the set of classes $\{\lambda_1, \lambda_2\}$ is the same as $\{\lambda_1, \lambda_3\}$ or $\{\lambda_2, \lambda_3\}$. This happens because, λ_1 is as different from λ_2 as λ_2 is from λ_3. However, in LR, distinct rankings can range from *completely different* to *very similar*. Considering these two sets:

$$1)\, \{(1,2,3,4,5)\,, (1,2,3,5,4)\}$$

$$2)\, \{(1,2,3,4,5)\,, (5,4,3,2,1)\}$$

and since the ranking proportions will be the same in 1) and 2), the entropy will be the same. Also, from a pairwise-comparison point of view, the two similar rankings in set 1) match 14 pairs from a total of 15, while the rankings in 2) do not match any.

Considering that entropy is a measure of disorder, we believe that it makes sense to expect lower entropy for sets with similar rankings and bigger entropy for sets with completely different rankings.

For this reason we propose to add an extra parameter in the formula of entropy for rankings (equation 4) to force lower values on sets of similar rankings. This means we have to adapt Shannon entropy for a set of rankings to be more sensitive to the similarity of the rankings present in the set.

$$Ent_{LR}\,(S) = \sum_{i=1}^{K} P\,(\pi_i, S)\, log\,(P\,(\pi_i, S))\, log\,(Q\,(\pi_i, S)) \tag{4}$$

where $Q\,(\pi_i, S)$ is the average similarity of the ranking π_i with the rankings in the subset S defined as:

$$Q\,(\pi_i, S) = \frac{\sum_{j=1}^{N} s\,(\pi_i, \pi_j)}{N} \tag{5}$$

For the same reason we find noise in independent variables, it is expected to observe the same phenomenon in ranking data. As the number of labels increases, we expect to observe it with more frequency, since the number of possible combinations for k labels grows to $k!$. As an example, instances 6, 7 and 8 in D_{ex} can correspond to observations of the same "real" ranking, say $(5, 4, 3, 2, 1)$, but with some noise.

This measure of entropy for rankings we propose here will make the discretization method more robust to noise present in the rankings. In order to support this statement we provide an analysis of the behavior of the method with induced and controlled noise.

The results presented do not include an analysis on partial orders. Given that, Kendall τ is a measure of the proportion of the concordant pairs of labels, this entropy measure can still work with partial orders, as long as there is at least one pairwise per instance.

4.2 MDLP for LR

MDLP [10] is a well known method used to discretize continuous attributes for classification learning. The method tries to maximize the information gain and considers all the classes of the data as completely distinct classes. For this reason, we believe that the latter, as is, is not suitable for datasets which have rankings instead of classes in the target.

MDLP measures the information gain of a given split point by comparing the values of entropy. For each split point considered, the entropy of the initial interval is compared with the weighted sum of the entropy of the two resulting intervals. Given an interval S:

$$Gain_{LR}\left(A,T;S\right) = Ent_{LR}\left(S\right) - \frac{|S_1|}{N} Ent_{LR}\left(S_1\right) - \frac{|S_2|}{N} Ent_{LR}\left(S_2\right)$$

Where $|S_1|$ and $|S_2|$ is the number of instances in the left side (S_1) and the number of instances in the right side (S_2) of the cut point T, respectively, in the attribute A.

After the adaptation of entropy for sets of rankings proposed in Section 4.1, Minimum Description Length Principle for Ranking data (MDLP-R) comes in a natural way. We only need to replace the entropy for rankings in the MDLP definition presented in [10], which we transcribe below:

MDLPC Criterion The partition induced by a cut point T for a set S of N examples is accepted iff

$$Gain_{LR}\left(A,T;S\right) > \frac{log_2\left(N-1\right)}{N} + \frac{\Delta_{LR}\left(A,T;S\right)}{N}$$

and it is rejected otherwise.

Where $\Delta_{LR}\left(A,T;S\right)$ is equal to:

$$log_2\left(3^K - 2\right) - \left[K Ent_{LR}\left(S\right) - K_1 Ent_{LR}\left(S_1\right) - K_2 Ent_{LR}\left(S_2\right)\right]$$

5 Experimental Results

Since what we are proposing is essentially a pre-processing method, the quality of its discretization is hard to measure in a direct way. For this reason, the experimental setup is divided in two parts. In the first part we present the results obtained from controlled artificial datasets that should give an indication whether the method is performing as expected. The second part shows results of the APRIORI-LR algorithm [7] run on datasets from the KEBI Data Repository at Philipps University of Marburg [3].

Table 3 compares the intervals discretized by the MDLP-R and MDLP in dataset D_{ex}. As expected, since there are eight distinct rankings, the RAC approach with MDLP for classification will see eight distinct classes and break the dataset into eight intervals. MDLP-R, however, can identify the similarities of rankings, and only breaks the dataset into two intervals.

Table 4 gives a description of the benchmark datasets from KEBI Data Repository at Philipps University of Marburg [3].

Table 3. Discretization results using the MDLP and MDLP-R methods

				Partitions	
TID	Att	π	λ	MDLP-R	MDLP
1	0.1	(1,2,4,3,5)	a	1	1
2	0.2	(1,2,3,4,5)	b	1	2
3	0.3	(2,1,3,4,5)	c	1	3
4	0.4	(1,3,2,4,5)	d	1	4
5	0.5	(1,2,3,5,4)	e	1	5
6	0.6	(5,4,3,1,2)	f	2	6
7	0.7	(4,5,3,2,1)	g	2	7
8	0.8	(5,3,4,2,1)	h	2	8

Table 4. Summary of the datasets

Datasets	type	#examples	#labels	#attributes
autorship	A	841	4	70
bodyfat	B	252	7	7
calhousing	B	20640	4	4
cpu-small	B	8192	5	6
elevators	B	16599	9	9
fried	B	40769	5	9
glass	A	214	6	9
housing	B	506	6	6
iris	A	150	3	4
segment	A	2310	7	18
stock	B	950	5	5
vehicle	A	846	4	18
vowel	A	528	11	10
wine	A	178	3	13
wisconsin	B	194	16	16

5.1 Results on Artificial Datasets

Results obtained with artificial datasets can give more insight about how the discretization method performs. The synthetic datasets presented in this section are variations of a simple one which has only two initial rankings $\pi_1 = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$ and $\pi_2 = (10, 9, 8, 7, 6, 5, 4, 3, 2, 1)$. To make it as simple as possible, it has only one independent variable which varies from 1 to 100. The first 60 instances are variations of π_1 and the remaining are variations of π_2.

In order to test the advantages of our method in comparison with the RAC approach, we intentionally introduced noise in the target rankings, by performing several swaps. Each swap is an inversion of two consecutive ranks in every ranking of the data. For each ranking the choice of the pairs to invert is random. Swaps will be done repeatedly, to obtain different levels of noise.

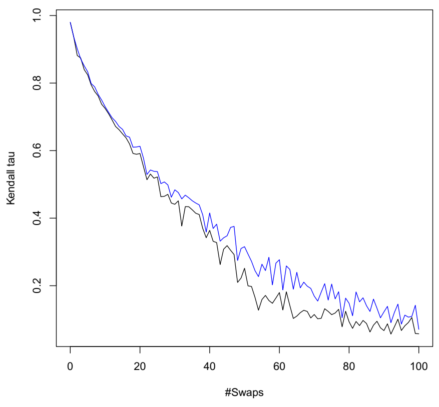


Fig. 1. Accuracy of the APRIORI-LR (expressed in terms of Kendall τ) as a function of the number of swaps, for MDLP (black) and MDLP-LR (blue)

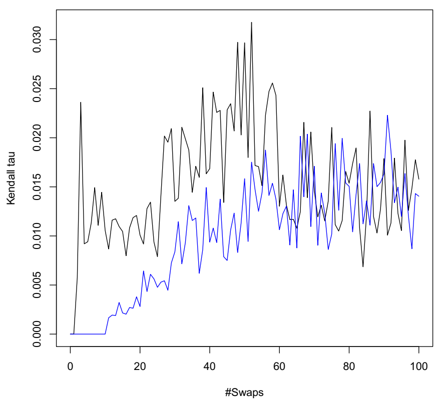


Fig. 2. Standard deviation of the accuracy of APRIORI-LR (in terms of Kendall τ) after discretization with MDLP-R (blue) and MDLP (black)

We performed an experiment which varies the number of swaps from 0 to 100. The greater the number of consecutive swaps, the more chaotic the dataset will be, and hence more difficult to discretize.

Figure 1 compares the accuracy of APRIORI-LR with two different discretization methods, MDLP and MDLP-R. The graph, clearly indicates that the discretization with MDLP-R (blue line) leads to better results for APRIORI-LR, than with MDLP after a RAC transformation . While for the first cases the difference is not so evident, as the noise increases, MDLP-R gives a greater contribution.

However, if we analyze Figure 2 there is extra information in favor of MDLP-R. The standard deviation of the 10 runs of the 10-fold cross-validation is zero in the presence of small amounts of noise (until approximately 10 swaps). This means that, in a scenario with reasonable noise in rankings, if one decides to use MDLP-R there are more chances to get the best result than with MDLP.

One great advantage of our method in this experiment can be seen in Figure 3. In particular, for any number of swaps until 20, our method only makes one partition which means that the split point choice is also invariant to a reasonable amount of noise. This will result in a small number of rules generated by APRIORI-LR as supported by the graph in Figure 4. In other words, MDLP-R makes APRIORI-LR much more efficient because it only needs to create approximately 1/10 of the rules to obtain the same accuracy.

In Figure 5 we can see the percentage of instances from the test set that were not ranked with the default rule. Since the minimum confidence was set to 50% from this graph we can conclude that ARIORI-LR with MDLP is decreasing the number of rules with confidence equal or higher than 50%. Thus, MDLP-R creates more meaningful intervals which lead to higher confidence LRAR.

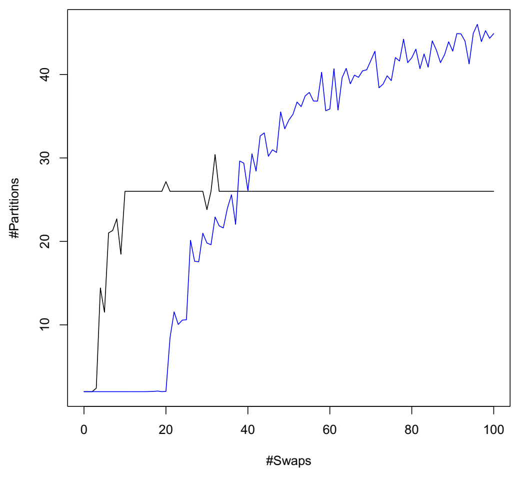


Fig. 3. Comparison of the average number of partitions generated by MDLP-R (blue) and MDLP (black)

Additionally, in Figure 3 it is clear that, from a certain point, MDLP is no longer able to make the distinction of the target attributes and so the average number of partitions stays constant. On the other hand, the number of partitions by MDLP-R increases as the noise increases too, which is an indicator that our method is able to perform discretization even in complex situations.

5.2 Results on Benchmark Datasets

The evaluation measure is Kendall's τ and the performance of the method was estimated using ten-fold cross-validation. For the generation of Label Ranking Association Rules (LR-AR) we used CAREN [2].

The similarity parameters used for the experiments are the same as used in [7]. The minimum support (*minsup*) was set to 0.1%. However, in some datasets, namely those with a larger number of attributes, frequent rule generation can be a very time consuming task. In this case, *minsup* was set to a larger value, 1% or higher.

When the algorithm cannot find at least one LRAR to rank a new instance, a default ranking is used. Since the usage of the default rule is only used as a last resourt, the minimum confidence (*minconf*) was adjusted with the same method used in [7] for parameter tunning. The latter aims to increase the percentage of test examples ranked without recourse to the default rule. This percentage is shown in column M in Table 5.

Table 5 shows that both methods obtain similar results in benchmark datasets. As we observed in the results from artificial datasets, the number of rules generated after a MDLP-R discretization is higher than with MDLP in more noisy/complex datasets. This is due to a higher number of partitions. The same

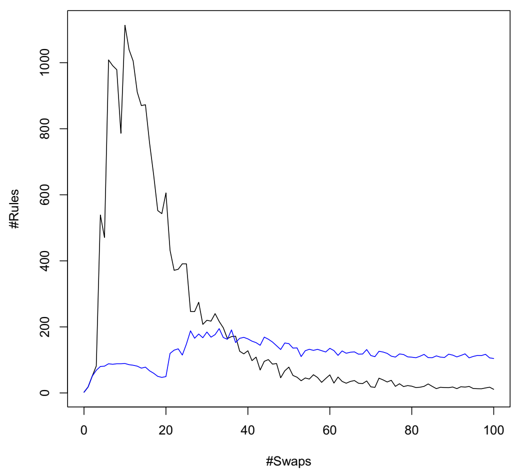


Fig. 4. Comparison of the number of rules generated by APRIORI-LR after discretization with MDLP-R (blue) and MDLP (black)

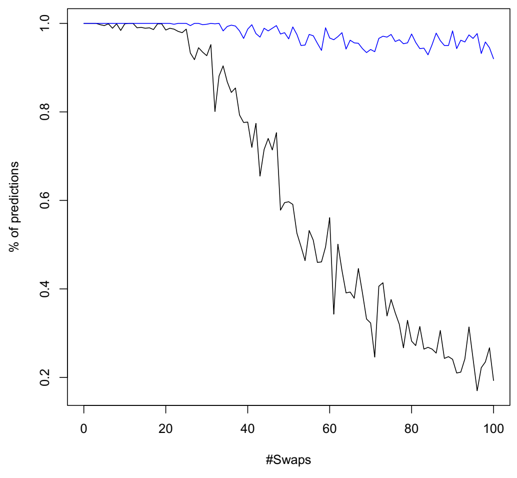


Fig. 5. Comparison of the percentage of instances from the test set that were not ranked with the default rule. MDLP-R (blue) MDLP (black).

phenomenon is also clear in Table 5 where the number of rules is generally higher with a MDLP-R discretization.

A baseline method uses the default LRAR, which is a rule with the average ranking, and the accuracy is presented in column $\tau_{baseline}$ as show in Table 5

Table 5. Results obtained with MDLP discretization and with *MDLP-R* discretization on bechmark datasets

	MDLP						MDLP-Ranking					
	τ	$\tau_{baseline}$	minsup	minconf	#rules	M	τ	$\tau_{baseline}$	minsup	minconf	#rules	M
authorship	.666	.568	15	90	14975	100%	.691	-568	1.5	100	6010	100%
bodyfat	.063	-.063	0.1	70	17135	100%	.066	-.063	0.1	65	23415	100%
calhousing	.329	.048	0.1	35	488	100%	.304	.048	0.1	30	1315	100%
cpu-small	.418	.234	0.1	35	326	100%	.458	.234	0.1	40	3888	100%
elevators	.648	.288	0.1	60	291	98%	.670	.288	0.1	60	5681	98%
fried	.802	-.002	0.1	55	8257	97%	.735	-.002	0.1	100	10445	100%
glass	.817	.684	0.1	80	168	100%	.860	.684	0.1	100	1686	100%
housing	.779	.053	0.1	65	420	100%	.809	.072	0.1	70	1284	100%
iris	.962	.089	0.1	85	36	100%	.934	.089	0.1	80	41	100%
segment	.898	.371	1	85	5110	100%	.890	.371	1	80	2888	100%
stock	.894	.072	0.1	80	1197	100%	.890	.072	0.1	80	2980	100%
vehicle	.847	.179	0.1	90	35085	100%	.835	.179	0.1	100	140051	100%
vowel	.802	.195	0.1	100	7593	100%	.632	.195	0.1	100	15606	100%
wine	.937	.329	0.1	100	1192	100%	.877	.329	0.1	100	3666	100%
wisconsin	.321	-.031	0.1	100	18223	100%	.271	-.031	0.1	100	16799	100%

6 Conclusions

In his paper we present a simple adaptation of the supervised discretization method, MDLP, for LR. This work was motivated by the lack of supervised discretization methods to deal with rankings in the target variable. The results clearly show that this is a viable LR method.

Our method clearly outperforms MDLP in the experiments with artificial data. In this work we empirically show that, in simple scenarios, MDLP-R deals with noisy ranking data accordingly to expected. Hence the latter is more reliable, in this kind of situation, than MDLP.

In Section 5.1 there are two sides of the new MDLP-R. In the presence of very simple LR problems (#swaps ≤ 20), it has less partitions and APRIORI-LR generates fewer rules than with MDLP. On the other hand, in more complex situations ($\#swaps > 20$) it has more partitions than MDLP and, consequently, APRIORI-LR creates more rules. The latter, in our opinion, cannot be seen as a disadvantage. The fact that there are more partitions being made means that the method is still able to identify similar groups of rankings even in very complex cases.

We believe that the measure of entropy for rankings proposed here, despite its heuristic nature, makes sense and its useful in the LR field. This new measure and MDLP-R bring new possibilities for processing ranking data and can motivate the creation of new methods for LR learning that cannot deal with continuous data. Furthermore, even though it was developed in the context of the LR task, it can be also applied to other fields such as regression since it is based on a distance measure such as Kendall τ.

This work uncovered several possibilities that could be better studied in order to improve the discretization in the LR field. They include: the choice of parameters in the stopping criterion; the usage of other entropy measures for rankings.

We also believe that it is essential to test the methods on real LR problems like metalearnig or predicting the rankings of financial analysts [1]. The KEBI datasets are adapted from UCI classification problems. In terms of real world applications, these can be adapted to rank analysts, based on their past performance and also preferences of radios, based on user's preferences.

Acknowledgments. This work was partially supported by Project Best-Case, which is co-financed by the North Portugal Regional Operational Programme (ON.2 - O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF).

We thank the anonymous referees for useful comments.

References

1. Aiguzhinov, A., Soares, C., Serra, A.P.: A similarity-based adaptation of naive bayes for label ranking: Application to the metalearning problem of algorithm recommendation. In: Pfahringer, B., Holmes, G., Hoffmann, A. (eds.) DS 2010. LNCS, vol. 6332, pp. 16–26. Springer, Heidelberg (2010)
2. Azevedo, P.J., Jorge, A.M.: Ensembles of jittered association rule classifiers. Data Min. Knowl. Discov. 21(1), 91–129 (2010)
3. Cheng, W., Hühn, J., Hüllermeier, E.: Decision tree and instance-based learning for label ranking. In: ICML 2009: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 161–168. ACM, New York (2009)
4. Cheng, W., Hüllermeier, E.: Label ranking with abstention: Predicting partial orders by thresholding probability distributions (extended abstract). CoRR, abs/1112.0508 (2011)
5. Cheng, W., Hüllermeier, E., Waegeman, W., Welker, V.: Label ranking with partial abstention based on thresholded probabilistic models. In: Advances in Neural Information Processing Systems 25, pp. 2510–2518 (2012)
6. Chiu, D.K.Y., Cheung, B., Wong, A.K.C.: Information synthesis based on hierarchical maximum entropy discretization. J. Exp. Theor. Artif. Intell. 2(2), 117–129 (1990)
7. de Sá, C.R., Soares, C., Jorge, A.M., Azevedo, P., Costa, J.: Mining association rules for label ranking. In: Huang, J.Z., Cao, L., Srivastava, J. (eds.) PAKDD 2011, Part II. LNCS, vol. 6635, pp. 432–443. Springer, Heidelberg (2011)
8. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: Machine Learning - International Workshop Then Conference, pp. 194–202 (1995)
9. Elomaa, T., Rousu, J.: Efficient multisplitting revisited: Optima-preserving elimination of partition candidates. Data Min. Knowl. Discov. 8(2), 97–126 (2004)
10. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: IJCAI, pp. 1022–1029 (1993)
11. Gurrieri, M., Siebert, X., Fortemps, P., Greco, S., Słowiński, R.: Label ranking: A new rule-based label ranking method. In: Greco, S., Bouchon-Meunier, B., Coletti, G., Fedrizzi, M., Matarazzo, B., Yager, R.R. (eds.) IPMU 2012, Part I. CCIS, vol. 297, pp. 613–623. Springer, Heidelberg (2012)
12. Kendall, M., Gibbons, J.: Rank correlation methods. Griffin, London (1970)
13. Kotsiantis, S., Kanellopoulos, D.: Discretization techniques: A recent survey. GESTS International Transactions on Computer Science and Engineering 32(1), 47–58 (2006)
14. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Knowledge Discovery and Data Mining, pp. 80–86 (1998)
15. Ribeiro, G., Duivesteijn, W., Soares, C., Knobbe, A.: Multilayer perceptron for label ranking. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part II. LNCS, vol. 7553, pp. 25–32. Springer, Heidelberg (2012)
16. Spearman, C.: The proof and measurement of association between two things. American Journal of Psychology 15, 72–101 (1904)

Identifying Super-Mediators of Information Diffusion in Social Networks

Kazumi Saito[1], Masahiro Kimura[2], Kouzou Ohara[3], and Hiroshi Motoda[4]

[1] School of Administration and Informatics, University of Shizuoka
k-saito@u-shizuoka-ken.ac.jp

[2] Department of Electronics and Informatics, Ryukoku University
kimura@rins.ryukoku.ac.jp

[3] Department of Integrated Information Technology, Aoyama Gakuin University
ohara@it.aoyama.ac.jp

[4] Institute of Scientific and Industrial Research, Osaka University
motoda@ar.sanken.osaka-u.ac.jp

Abstract. We propose a method to discover a different kind of influential nodes in a social network, which we call "super-mediators", *i.e.*, those nodes which play an important role in receiving the information and passing it to other nodes. We mathematically formulate this as a difference maximization problem in the average influence degree with respect to a node removal, *i.e.*, a node that contributes to making the difference large is influential. We further characterize the property of these super-mediators as having both large influence degree, *i.e.*, capable of widely spreading information to other recipient nodes, and large reverse-influence degree, *i.e.*, capable of widely receiving information from other information source nodes. We conducted extensive experiments using three real world social networks and confirmed that this property holds. We further investigated how well the conventional centrality measures capture super-mediators. In short the in-degree centrality is a good measure when the diffusion probability is small and the betweenness centrality is a good measure when the diffusion probability is large, but the super-mediators do depend on the value of the diffusion probability and no single centrality measure works equally well for a wide range of the diffusion probability.

Keywords: Information diffusion, super-mediator, influence degree, reverse-influence degree.

1 Introduction

The emergence of Social Media such as Facebook, Digg and Twitter has provided us with the opportunity to create large social networks, which play a fundamental role in the spread of information, ideas, and influence. Such effects have been observed in real life, when an idea or an action gains sudden widespread popularity through "word-of-mouth" or "viral marketing" effects. This phenomenon has attracted the interest of many researchers from diverse fields [12], such as sociology, psychology, economy, computer science, etc.

A substantial amount of work has been devoted to the task of analyzing and mining information diffusion (*i.e.*, cascading) processes in large social networks

[16,14,2,1,18,25,3]. Widely used information diffusion models in these studies are *independent cascade (IC)* [4,6,7], *linear threshold (LT)* [26,27] and their variants [8,19,5,10,20,21]. These two models focus on different aspects of information diffusion. IC model is sender-centered (information push) and each active node *independently* influences its inactive neighbors with given diffusion probabilities. LT model is receiver-centered (information pull) and a node is influenced by its active neighbors if their total weight exceeds the threshold for the node. Basically the former models diffusion process of how a disease spreads and the latter models diffusion process of how an opinion or innovation spreads.

The main focus of research using these models over the past decade has been on optimization problems in which the goal is to maximize the spread of information through a given network, either by selecting a good subset of nodes to initiate the cascade [6,9] or by applying a broader set of intervention strategies such as node and link additions [17,24]. In particular the former problem is well studied as the *influence maximization problem*, *i.e.*, finding a subset of nodes of size K that maximizes the expected influence degree with K as a parameter. In [23] we proposed a new type of influence maximization problem which we called "Target selection problem" (to avoid confusion, we called the original influence maximization problem as "Source selection problem"). The difference is that the new problem does not assume that the information is guaranteed to start spreading from the selected target nodes. Rather we send the same information from outside of the network to the selected targets as a probabilistic process. This is closer to a situation in which we send a direct mail to selected customers expecting that they spread the received information to others. What we found very interesting is that the nodes selected as the solution of the target selection problem were substantially different from the nodes selected as the solution of the source selection problem, especially in case of LT model. We attributed the difference to the fact that the target nodes must not only be able to be influential, *i.e.*, capable of widely spreading information to other recipients, but also be able to be reverse-influential, *i.e.*, capable of widely receiving the information from other sources. In a separate context we studied another type of influential nodes which we called "super-mediators", *i.e.*, nodes which play an important role in receiving the information and passing it to other nodes [22]. There, we empirically[1] defined the super-mediators to be the nodes that frequently appear in the long information diffusion sequences that start from a node and are shared by many of these sequences that starts from different nodes[2]. The biggest difference of the present work from [22] is that the present work is model-driven while our previous work is data-driven, *i.e.*, [22] does not assume any diffusion model but it requires that abundant observed diffusion sequences are available.

The work in this paper is motivated by these studies. "Source selection problem" only cares the ability of nodes to spread information. "Target selection problem" cares also the ability of the nodes to receive information in addition to the ability to spread the information, but only in the first step of information diffusion chain. Super-mediators share the same concept as "target selection problem", but they can be any nodes in the

[1] We call it empirical in the sense that the characterization is qualitative and there is no mathematically defined objective function to be optimized.

[2] We assume that there are many sequences of different length for each starting node.

chain of information diffusion process. From this observation, we can mathematically define the super-mediators as the solution of an optimization problem and rank them. The influence degree $\sigma(v)$ of a node v is defined to be the expected number of active nodes at the end of diffusion process, *i.e.*, nodes that have become influenced due to information diffusion (see Sec.2 for a more rigorous definition). The average influence degree of the whole network is defined to be the average of $\sigma(v)$ over all nodes in the network. If a node v is a super-mediator, removing this node would substantially decrease the average influence degree. Thus, the importance of each node as a super-mediator can be quantified as the difference of the average influence degree with respect to the node removal.

Here in this paper we use IC model as the information diffusion model and only consider a single node removal, *i.e.*, $K = 1$, but this optimization problem carries the same problem of computational complexity of estimating influence degree[3]. We devised the bond percolation [9] and pruning [8] algorithms to efficiently estimate the influence degree. In this paper, we further improved these techniques and reduced the computation time drastically (but this is not our focus in this paper).

We wanted to characterize the property of the super-mediators returned as the solution of the optimization problem. As mentioned above, there are two important factors: the ability to spread information and the ability to receive information. The former is captured by the influence degree. The latter is captured by the reverse-influence degree, which is a new concept born in this study, *i.e.*, the expected number of initial source nodes from which the information reaches a node at the end of information diffusion. Our hypothesis is that the super-mediators should be ranked high in terms of both of them. We have tested our hypothesis using three real world networks (Enron, Blog and Wikipedia), and confirmed that this property holds. In case of Enron e-mail network, the nodes identified as super-mediators are interpretable in the light of open literature. We further investigated whether the conventional centrality measures can serve as a good measure to identify the super-mediators. What we found is that the super-mediators depend on the value of the diffusion probability and in short the in-degree centrality is a good measure when the diffusion probability is small and the betweenness centrality is a good measure when the diffusion probability is large, and no single centrality measure works equally well for a wide range of the diffusion probability. It can be said that the measure we proposed in this paper is a new centrality that can be added to the existing pool, but the difference is that this measure explicitly considers information diffusion process while the existing centrality considers only network structure.

The paper is organized as follows. Section 2 gives a brief description of the independent cascade model. Section 3 defines super-mediators and gives an algorithm to find and rank them. Section 4 characterizes the super-mediators and introduces a new concept "reverse influence degree" and gives an efficient way to compute it. Section 5 reports experimental results and shows that the hypothesis we made holds for the three networks. Section 6 summarizes what has been achieved in this work and addresses the future work.

[3] If we consider $K > 1$, the problem becomes more difficult, but we can still use the sub modular property and the same greedy algorithm as is used in "Source selection problem" with various tactics, *e.g.*, burnout [19].

2 Information Diffusion Model

We consider a network represented by a directed graph $G = (V, E)$, where V and E ($\subset V \times V$) are the sets of all the nodes and links, respectively. Below we revisit the definition of IC model according to the literatures [6,11]. The diffusion process proceeds from an initial active node in discrete time-step $t \geq 0$, and it is assumed that nodes can switch their states only from inactive to active (*i.e.*, the SIR setting, see Section 3).

IC model has a *diffusion probability* $p_{u,v}$ with $0 < p_{u,v} < 1$ for each link (u, v) as a parameter. Suppose that a node u first becomes active at time-step t, it is given a single chance to activate each currently inactive child node v, and succeeds with probability $p_{u,v}$. If u succeeds, then v will become active at time-step $t + 1$. If multiple parent nodes of v first become active at time-step t, then their activation trials are sequenced in an arbitrary order, but all performed at time-step t. Whether u succeeds or not, it cannot make any further trials to activate v in subsequent rounds. The process terminates if no more activations are possible.

For an initial active node $v \in V$, let $\varphi(v; G)$ denote the number of active nodes at the end of the random diffusion process. It is noted that $\varphi(v; G)$ is a random variable. We denote the expected value of $\varphi(v; G)$ by $\sigma(v; G)$, and call it the *influence degree of* v.

3 Identifying Super-Mediators

As stated earlier, we conjecture that if a node w is a super-mediator, removing this node would substantially decrease the average influence degree. In order to mathematically formulate this notion, we first define the following graph $G \setminus \{w\}$, which is constructed by removing a node w from a directed graph $G = (V, E)$:

$$G \setminus \{w\} = (V \setminus \{w\}, E \setminus \{w\}), \quad E \setminus \{w\} = \{(u, v) \in E \mid u \neq w, v \neq w\}. \tag{1}$$

Then, we can quantify the super-mediator degree of each node w, denoted by $medt(w)$, as the difference in the average influence degree with respect to the node removal, *i.e.*,

$$medt(w) = \sum_{v \in V} \sigma(v; G)p(v) - \sum_{v \in V \setminus \{w\}} \sigma(v; G \setminus \{w\})p(v), \tag{2}$$

where $p(v)$ stands for the probability that the node v becomes *an information source node*, that is, an initial active node. Of course, we want to identify the nodes that have large values of super-mediator degree.

We apply our bond percolation technique [9] to efficiently calculate the super-mediator degree $medt(w)$ for each node $w \in V$. Note first that the IC model on G can be identified with the so-called susceptible/infective/recovered (SIR) model [15,27] for the spread of a disease on G, where the nodes that become active at time t in the IC model correspond to the infective nodes at time t in the SIR model. Recall that in the SIR model, each individual occupies one of the three states, "susceptible", "infected" and "recovered", where a susceptible individual becomes infected with a certain probability when it encounters an infected patient, and subsequently recovers at a certain rate. It is known that the SIR model on a network can be exactly mapped onto a bond percolation model on the same network [15,6]. Thus, the IC model on G is equivalent to a bond percolation model on G, that is, these two models have the same probability distribution for the final set of active nodes. Our bond percolation technique [9] exploits this relationship. Here, we present the algorithm for calculating $medt(w)$ based on the bond percolation technique. A bond percolation process on G is the process in which each link of G is randomly designated either "occupied " or "unoccupied" according to some probability distribution in which the occupation probability over each link (u, v) is set to the diffusion probability $p_{u,v}$. Now, we consider M times of bond percolation processes. Let E_m denote the set of occupied links at the m-th bond percolation process and let G_m denote the graph (V, E_m), then for a large M, we can calculate the estimated influence degree $\bar{\sigma}(u; G)$ with a reasonable accuracy as follows:

$$\bar{\sigma}(u; G) = \frac{1}{M} \sum_{m=1}^{M} |R(u; G_m)|, \tag{3}$$

where $R(u; G_m)$ stands for a set of reachable nodes from u on G_m such that there is a path from u to v for $v \in R(u; G_m)$, and $|R(u; G_m)|$ is the number of nodes in $R(u; G_m)$. Here note that our bond percolation technique decomposes each graph G_m into its SCCs, where SCC (strongly connected component) is a maximal subset C of V such that for all u, $v \in C$ there is a path from u to v. Namely, $R(u; G_m) = R(v; G_m)$ if u, $v \in C$. Thus, we can obtain $R(u; G_m)$ for any node $u \in V$ by calculating $R(u; G_m)$ for only one node u in each component C.

We obtain the following estimation formula by substituting Equation (3) into Equation (2):

$$medt(w) = \frac{1}{M} \sum_{v \in V} \sum_{m=1}^{M} |R(v; G_m)| p(v) - \frac{1}{M} \sum_{v \in V \setminus \{w\}} \sum_{m=1}^{M} |R(v; G_m \setminus \{w\})| p(v). \tag{4}$$

In order to efficiently calculate $R(v; G_m \setminus \{w\})$ for each pair of nodes, v and w, we consider a set of reverse reachable nodes defined by

$$R^{-}(w; G_m) = \{v \in V \mid w \in R(v; G_m)\}.$$

Then, we can easily see that

$$v \notin R^{-}(w; G_m) \implies R(v; G_m \setminus \{w\}) = R(v; G_m).$$

Namely, for the m-th bond percolation process and a fixed node w, we can obtain $R(v; G_m \setminus \{w\})$ for any node $v \in V$ by calculating $R(v; G_m \setminus \{w\})$ only for $v \in R^{-}(w; G_m)$. Here, as described above, we can further improve the efficiency by applying SCC decomposition for a subgraph consisting of nodes in $R^{-}(w; G_m)$. Below we can summarize our proposed algorithm for calculating the super-mediator degree $medt(w)$ for each node $w \in V$.

1. Perform bond percolation process M times ($m = 1, \cdots, M$);
   (a) For the m-th bond percolation process, calculate $R(v; G_m)$ by applying SCC decomposition;

(b) For each $w \in V$, compute $R^{-}(w; G_m)$, and for each $v \in V$, set $R(v; G_m \setminus \{w\}) = R(v; G_m)$ if $v \notin R^{-}(w; G_m)$; otherwise calculate $R(v; G_m \setminus \{w\})$ by applying SCC decomposition;

2. Calculate the super-mediator degree $medt(w)$ according to Equation (4).

4 Characterizing Super-Mediator

As mentioned earlier, we attempt to characterize the property of the super-mediators by two important factors for each node v: the influence degree $\sigma(v; G)$ and the *reverse-influence degree* denoted by $\sigma^{-}(v; G)$. First of all, in order to quantify the relationships between these two factors, we define the probability $\sigma(u, v; G)$ that the node v becomes active when u is an information source node. Then, we can calculate $\sigma(v; G)$ by $\sum_{u \in V} \sigma(v, u; G)$. On the other hand, if we define the *reverse-influence degree* as the expected number of initial source nodes from which the information reaches the node v at the end of information diffusion, we can define $\sigma^{-}(v; G)$ by

$$\sigma^{-}(v; G) = \sum_{u \in V} \sigma(u, v; G).$$

In order to further quantify the relationships, we consider the following reverse graph G^{-}, which is constructed by reversing any link $(u, v) \in E$ for a directed graph $G = (V, E)$.

$$G^{-} = (V, E^{-}), \quad E^{-} = \{(v, u) \mid (u, v) \in E\}. \tag{5}$$

Then, we can show that the reverse-influence degree of each node v is equal to the influence degree of node v on the reverse graph G^{-}, *i.e.*,

$$\sigma^{-}(v; G) = \sigma(v; G^{-}). \tag{6}$$

To confirm this fact, we introduce a function $R(u, v; G_m)$ of $v \in V$ such that $R(u, v; G_m) = 1$ if there is a path from u to v on G_m, and $R(u, v; G_m) = 0$ otherwise, where G_m is the graph obtained by the m-th bond percolation process in Section 3. Noting that $R(u, v; G_m) = R(v, u; G_m^{-})$, it is straightforward to show that Equation (6) holds as shown below:

$$\begin{aligned} \bar{\sigma}^{-}(v; G) &= \frac{1}{M} \sum_{m=1}^{M} \sum_{u \in V} R(u, v; G_m) \\ &= \frac{1}{M} \sum_{m=1}^{M} \sum_{u \in V} R(v, u; G_m^{-}) \\ &= \bar{\sigma}(v; G^{-}), \end{aligned} \tag{7}$$

where G_m^{-} is the reverse graph of G_m. As a natural conjecture, we can expect that the super-mediator nodes are influential on both a given graph G and its associated reverse graph G^{-}, which respectively corresponds to the influence degree $\sigma(v; G)$ and the reverse-influence degree $\sigma^{-}(v; G)$. Thus, our hypothesis is that the super-mediators should be ranked high in terms of both of them. We empirically evaluate this hypothesis using three real world networks since exploring this analytically seems difficult.

5 Experiments

5.1 Datasets and Settings

We employed three datasets of large real networks. The first one is the Enron network, which is derived from the Enron Email Dataset [13]. We regarded each email address as a node, and constructed a link from email address u to email address v only if u sent an email to v. The Enron network is a directed network which has $19,603$ nodes and $210,950$ directed links. The second one is the Blog network, which is a trackback network of Japanese blogs used by Kimura et al [11]. The Blog network is also a directed network which has $12,047$ nodes and $53,315$ directed links. The third one is the Wikipedia network, which is a network of people derived from the "list of people" within Japanese Wikipedia, also used by Kimura et al [11]. The Wikipedia network is a bidirectional network having $9,481$ nodes and $245,044$ directed links.

Below we explain the parameter settings of IC model. We first assume a generative model according to the beta distribution with a mean of μ for the diffusion probability $p_{v,w}$ for any link $(v,w) \in E$. Note that the beta distribution is the conjugate prior probability distribution for the Bernoulli distribution corresponding to a single toss of a coin. We further suppose that each diffusion probability is independently generated from the beta distribution with respect to each information diffusion process. Then the average occupied probability of the bond percolation process over each link reduces to μ. Actually, this formulation is equivalent to assigning a uniform value μ to the diffusion probability $p_{v,w}$ for any link $(v,w) \in E$, that is, $p_{v,w} = \mu$. According to [6], we set the value of μ to a value that is less than or equal to $1/\bar{d}$, where $\bar{d}$ is the mean out-degree of a network. Thus, we investigate $\mu = r/\bar{d}$, where r is a parameter with $0 < r \leq 1$. The parameter M to estimate the expectation is set to 10,000 for all experiments. The probability that the node v becomes an information source node was assumed to be uniform, *i.e.*, $p(v) = 1/|V|$.

5.2 Centralities

We also investigated whether or not super-mediators can be identified by heuristic methods based on the three well-known centrality measures, *degree centrality*, *closeness centrality*, and *betweenness centrality* that are commonly used as the influence measure in sociology. Let $G = (V, E)$ be a directed network for our analysis, and let $G^- = (V, E^-)$ be the reverse network of G. For the degree centrality, we consider the *out-degree* of node $v \in V$, $deg^+(v)$, defined as the number of links from v, and the *in-degree* of node $v \in V$, $deg^-(v)$, defined as the number of links to v; i.e.,

$$deg^+(v) = |\{(v,w) \in E\}|, \quad deg^-(v) = |\{(w,v) \in E\}| = |\{(v,w) \in E^-\}|.$$

For the closeness centrality, we consider the *closeness* of node $v \in V$, $close(v)$, defined as

$$close(v) = \frac{1}{|V|} \sum_{w \in V} \frac{1}{dist(v,w)},$$

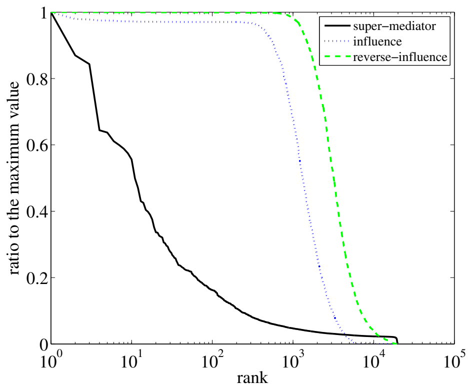


Fig. 1. Distribution of the super-mediator/influence/reverse-influence degree for the Enron network in case of $r = 1.0$

Table 1. Top 3 email accounts (nodes) in the super-mediator, influence, and reverse-influence degree ranking for the Enron network ($r = 1.0$)

	account name (ID: ratio to the maximum degree value)		
rank	super-mediator	influence	reverse-influence
1	jeff.skilling (642: 1.000)	bob.ambrocik (16734: 1.000)	tom.alonso (5510: 1.000)
2	kenneth.lay (471: 0.870)	technology.enron (17219: 0.979)	jeff.richter (1768: 0.999)
3	sally.beck (535: 0.843)	outlook.team (10779: 0.978)	chris.mallory (5933: 0.999)

and the *reverse closeness* of node $v \in V$, $close^{-}(v)$, defined as

$$close^{-}(v) = \frac{1}{|V|} \sum_{w \in V} \frac{1}{dist^{-}(v, w)},$$

where $dist(v, w)$ stands for the graph distance (shortest path length) from node v to node w in the network G, and $dist^{-}(v, w)$ stands for the graph distance from node v to node w in the reverse network G^{-}. For the betweenness centrality, we consider the *betweenness* of node $v \in V$, $betw(v)$, defined as the total number of shortest paths between pairs of nodes that pass through v. We consider detecting super-mediators by ranking the nodes in decreasing order with respect to a centrality measure. We refer to the detection methods by centrality measures $deg(v)$, $deg^{-}(v)$, $close(v)$, $close^{-}(v)$, and $betw(v)$ as the *out-degree*, *in-degree*, *closeness*, *reverse closeness*, and *betweenness* methods, respectively.

5.3 Results

Confirmation of Properties of Super-Mediators. First, we investigated the distributions of the three measures, *i.e.*, the super-mediator, influence, and reverse-influence degree in the Enron network to see how much they differ to each other in terms of characterizing each node. In Fig. 1, the values of “ratio to the maximum value” in each

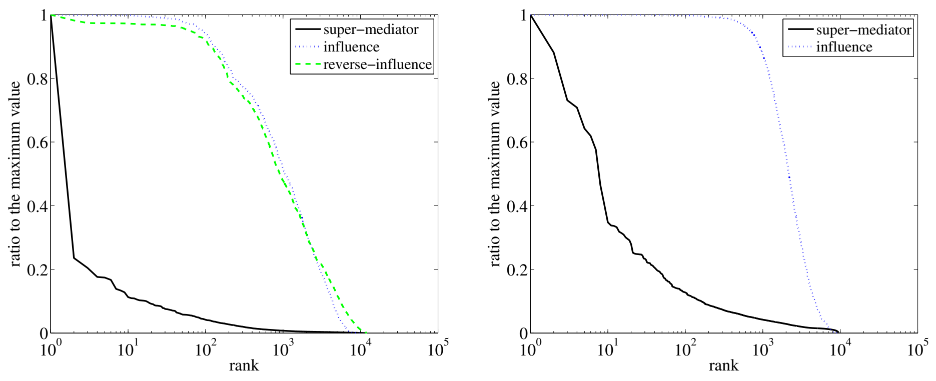
Table 2. The rank of the top 3 super-mediators for the Enron network in the influence and reverse-influence degree ranking (r = 1.0)

	rank (ratio to the maximum degree value)		
ID	super-mediator	influence	reverse-influence
642	1 (1.000)	441 (0.947)	642 (0.999)
471	2 (0.870)	122 (0.970)	374 (0.999)
535	3 (0.843)	126 (0.970)	377 (0.999)

degree are depicted as a function of node rank. Note that node rank is different for each degree. It can be observed that the curves for the influence and reverse-influence degree are similar to each other, while the curve for the super-mediator degree is quite different from the other two. Each curve is almost flat for the first two. The one for the influence degree maintains a relatively high ratio close to 1.0 up to approximately top 300 nodes and the one for the reverse-influence degree up to approximately top 1,000 nodes. This means that there is very little difference among these top ranked nodes as far as the influence is concerned. On the other hand, the distribution curve rapidly decreases to the top 1,000 nodes for the super-mediator degree. We can conclude that the super-mediator ranking can characterize each node by far clearly than the influence and reverse-influence ranking.

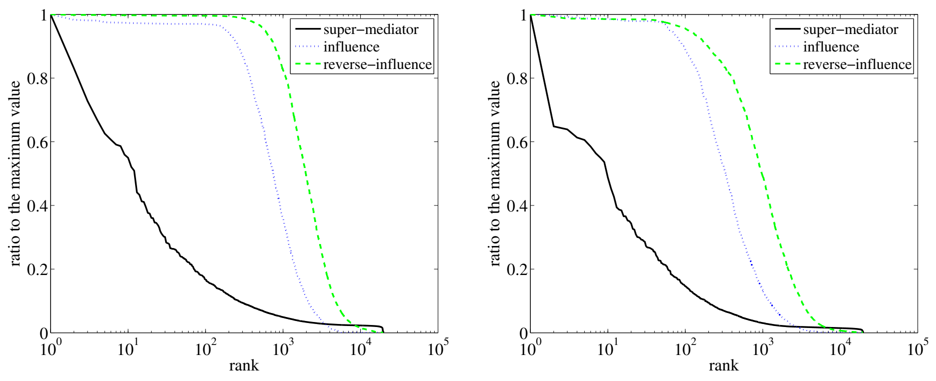
We further examined the top 3 nodes in each ranking for the Enron network in case of r = 1.0, and summarized them in Table 1. Again, we can observe that there is a clear difference in the values of the super-mediator degree among the top 3 email accounts (nodes), but the difference is not clear for the other two degree, especially the reverse-influence degree. In addition, these top 3 ranked super-mediators are different from the top 3 ranked nodes for the other two: the influence degree and the reverse-influence degree. It is notable that "Jeffrey Skilling" (the top ranked) and "Kenneth Lay" (the second ranked) in the super-mediator degree are key persons of the Enron scandal: "Jeffrey Skilling" is the former president of Enron and "Kenneth Lay" was the CEO of Enron. Both of them do not appear in the top 3 in both the influence and reverse-influence degree ranking. "Jeffrey Richter", the second ranked in the reverse-influence degree, is known as a trader of Enron, but is not as well-known as the former two executives. These observations suggest the super-mediator degree can be a promising measure to identify nodes that actually play an important role in a given network.

Next, we investigated how the top 3 super-mediators for the Enron network rank in terms of the influence and the reverse-influence degree values. Our conjecture is that the super-mediators should be ranked high in these two measures. The results are summarized in Table 2. It is found that these super-mediators are ranked relatively high, at least they are in the top 5% nodes. This confirms our conjecture. However, because their curves are flat, there are many other nodes that are ranked high in these two measures. This means that the reverse is not necessarily true, *i.e.*, super-mediators have high values for these two measures but having high values for these two measures are not necessarily super-mediators as defined in Equation (2). Indeed, we observed the same tendencies for the Blog and Wikipedia networks. Here, we show only the distributions of the three measures for these networks in case of r = 1.0 in Figs. 2a and 2b,



(a) Blog network (b) Wikipedia network

Fig. 2. Distribution of the super-mediator/influence/reverse-influence degree for the Blog and Wikipedia networks in case of $r = 1.0$



(a) $r = 0.5$ (b) $r = 0.25$

Fig. 3. Distribution of the super-mediator/influence/reverse-influence degree for the Enron network in cases of $r = 0.5$ and $r = 0.25$

respectively. Note that since the Wikipedia network is bidirectional, the reverse-influence degree is equivalent to the influence degree, so it is not shown in Fig. 2b.

Further Analysis. Using the Enron network, we further analyzed the properties of super-mediators. First, we investigated the effect of diffusion probability by varying the value of r. Figures 3a and 3b, and Tables 3 and 4 show the results for the case of $r = 0.5$ and $r = 0.25$, respectively. Here, each table indicates ranks and the values of "ratio to the maximum value" with respect to super-mediator, influence and reverse influence degree for the top 3 super-mediators. It is obvious that the distribution curves shown in Fig. 3a and 3b share the same tendency as those in Fig. 1. The notable difference is that the flat region of each curve shrinks for the influence and reverse-influence degree as the diffusion probability becomes smaller. This is because both $\sigma(v; G)$ and $\sigma^{-}(v; G)$

Table 3. The rank of the top 3 super-mediators for the Enron network in the influence and reverse-influence degree ranking (r = 0.5)

	rank (ratio to the maximum degree value)		
ID	super-mediator	influence	reverse-influence
535	1 (1.000)	66 (0.970)	94 (0.996)
471	2 (0.831)	114 (0.969)	128 (0.995)
642	3 (0.728)	426 (0.742)	341 (0.985)

Table 4. The rank of the top 3 super-mediators for the Enron network in the influence and reverse-influence degree ranking (r = 0.25)

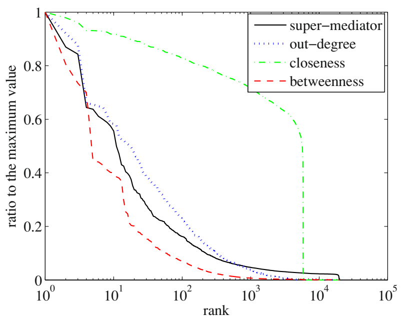
	rank (ratio to the maximum degree value)		
ID	super-mediator	influence	reverse-influence
535	1 (1.000)	52 (0.970)	46 (0.979)
6	2 (0.648)	185 (0.734)	1 (1.000)
471	3 (0.639)	154 (0.799)	144 (0.931)

become smaller for every node v in accordance with the decrease of the diffusion probability. Also from Tables 3 and 4, we can see the same tendency as for the case of $r = 1.0$ although the top 3 nodes and their rankings for $r = 0.5$ and $r = 0.25$ are not exactly the same as for $r = 1.0$. Further we notice that all the values for the influence and reverse-influence degree are not very high due to the aforementioned shrink of the flat region, *i.e.*, third rank for $r = 0.5$ and the second and the third rank for $r = 0.25$, but overall both the influence degree and the reverse-influence degree are high for the high ranked super-mediators. Indeed, in the influence and reverse-influence ranking, these nodes are within the top 3% nodes at $r = 0.5$, and within the top 1% nodes at $r = 0.25$.

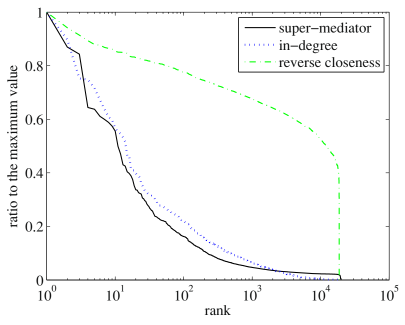
Next, we investigated whether the conventional centrality measures can serve as a good measure to identify the super-mediators. Figure 4 displays the values of "ratio to the maximum value" as a function of node rank with respect to out-degree $deg^{+}(v)$, in-degree $deg^{-}(v)$, closeness $close(v)$, reverse closeness $close^{-}(v)$, and betweenness $betw(v)$. We observe that the distributions of out-degree $deg^{+}(v)$, in-degree $deg^{-}(v)$ and betweenness $betw(v)$ are similar to the distribution of the super-mediator degree, while the distributions of closeness $close(v)$ and reverse closeness $close^{-}(v)$ are similar to the distributions of the influence and reverse-influence degree. Here note that the value of "ratio to the maximum value" of a node with respect to the super-mediator degree is less than 0.2 for nodes ranked after the top 100. Thus, we focused on the top 100 nodes, and examined the similarity between the super-mediator ranking and the other ranking, *i.e.*, the out-degree, in-degree, closeness, reverse closeness, and betweenness ranking. Here, we measured the similarity between the top k nodes for one ranking method, denoted as a set A_k, and those for the other ranking method, denoted as a set A'_k, by the F-measure $F(k)$ defined by

$$F(k) = \frac{|A_k \cap A'_k|}{k}.$$

Figures 5a, 5b and 5c show the results for the cases of $r = 1.0$, $r = 0.5$ and $r = 0.25$, respectively. Figure 5d displays the similarities between the super-mediator ranking

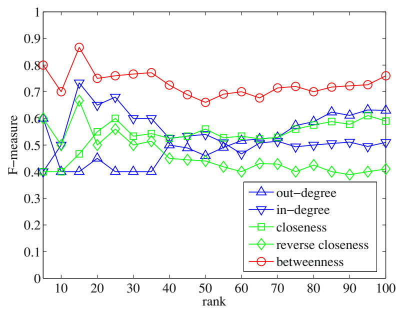


(a) Out-degree, closeness, and betweenness

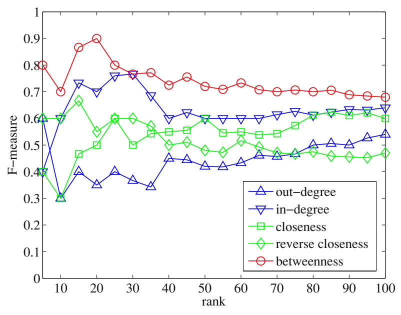


(b) In-degree and reverse closeness

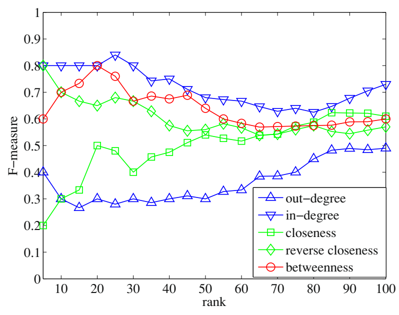
Fig. 4. Distributions of conventional centrality measures for the Enron network



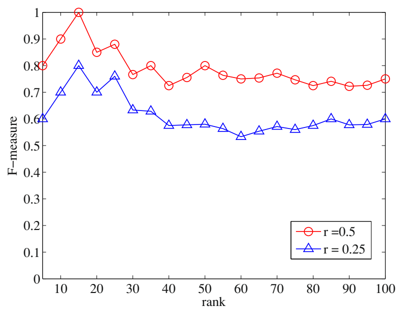
(a) $r = 1$



(b) $r = 0.5$



(c) $r = 0.25$



(d) Comparison of super-mediators with r

Fig. 5. Relation between conventional centrality and super-mediator degree for the Enron network

for the case of $r = 1.0$ and that of $r = 0.5$ and $r = 0.25$. We notice that the super-mediators depend on the value of the diffusion probability from Fig. 5d. We further notice that the betweenness centrality is best when the diffusion probability is large (Fig. 5a, 5b) and the in-degree centrality becomes better when the diffusion probability gets smaller (Fig. 5c). It is interesting that the out-degree centrality is not as good as

Table 5. Top 3 nodes for conventional centrality measures for the Enron network for $r = 1.0$

rank	out-degree	in-degree	closeness	reverse-closeness	betweenness
1	451	6	535	6	6
2	10779	203	10779	203	642
3	535	535	451	684	471

the in-degree centrality. Further investigation is needed to understand this phenomenon. Table 5 shows the top 3 nodes for the out-degree, in-degree, closeness, reverse-closeness, and betweenness centrality in case of $r = 1.0$. These should be compared with the node IDs in Table 2, *i.e.*, 642, 471 and 535. Two nodes (642, 471) for the betweenness centrality match them and one node (535) for the out-degree, in-degree and closeness centrality matches them. This supports the above observation. In summary no single centrality measure works equally well for a wide range of the diffusion probability. The betweenness centrality is a good measure when the diffusion probability is large and in-degree centrality is a good measure when the diffusion probability is small. This is intuitively understandable. When the diffusion probability is large, there are many long diffusion sequences, in which case the betweenness plays a key role, whereas the diffusion probability is small, many of the diffusion sequences are short, in which case node degree plays a key role.

6 Conclusion

We addressed a problem of identifying and characterizing influential nodes in a social network which we call "super-mediators" (nodes which play a role of mediator), *i.e.*, nodes that play an important role in receiving the information and passing it to other nodes. This notion of influential nodes is different from the conventional one in which a node is said to be influential if the information starting from that node spreads to many other nodes. We quantified the degree of importance as a super-mediator degree and formulated this as the difference of the average influence degree with respect to the node removal. If a node is a super-mediator, removal of this node from the network will substantially decrease the average influence degree. Thus finding the most influential super-mediator is finding a node that maximizes this difference. We can rank the super-mediators according to the amount of difference. This computation requires to estimate influence degree of each node, which is defined to be the expected number of active nodes at the end of information diffusion process, and is very time consuming. We used our bond percolation approach to simulate an individual diffusion process and the expectation is approximated by the empirical mean of many trials of diffusion process. We conjectured that super-mediators would have both large influence degree, *i.e.*, capable of widely spreading information to other recipient nodes, and large reverse-influence degree, *i.e.*, capable of widely receiving information from other information source nodes. In fact reverse-influence degree of a node in a graph is the same as the influence degree of the same node of the graph in which the edge direction is reversed for all edges. We conducted extensive experiments using three real world social networks

(Enron, Blog and Wikipedia) with different diffusion probability assuming independent cascade model, and confirmed that this conjecture is correct, but the reverse is not correct, *i.e.*, nodes that have both large influence degree and large reverse-influence degree are not necessarily super-mediators. The performance of super-mediator degree is tested in the Enron network. The top three super-mediators identified by our method are confirmed to be actually influential. We further investigated how well the conventional centrality measures (in-degree, out-degree, closeness, reverse-closeness and betweenness) capture super-mediators. In short the in-degree centrality is a good measure when the diffusion probability is small and the betweenness centrality is a good measure when the diffusion probability is large, but the super-mediators do depend on the value of the diffusion probability and no single centrality measure works equally well for a wide range of the diffusion probability. Our immediate future work is to investigate the generality of the findings reported in this paper for a variety of networks and elucidate why the out-degree centrality is not as good a measure as the in-degree centrality.

Acknowledgments. This work was partly supported by Asian Office of Aerospace Research and Development, Air Force Office of Scientific Research under Grant No. AOARD-13-4042, and JSPS Grant-in-Aid for Scientific Research (C) (No. 23500194).

References

1. Bakshy, E., Hofman, J., Mason, W., Watts, D.: Everyone's an influencer: Quantifying influences on twitter. In: Proceedings of the 4th International Conference on Web Search and Data Mining (WSDM 2011), pp. 65–74 (2011)
2. Bakshy, E., Karrer, B., Adamic, L.A.: Social influence and the diffusion of user-created content. In: Proceedings of the 10th ACM Conference on Electronic Commerce, pp. 325–334 (2009)
3. Dow, P., Adamic, L., Friggeri, A.: The anatomy of large facebook cascades. In: Proceedings of the 7th International AAAI Conference on Weblogs and Social Media, ICWSM (2013)
4. Goldenberg, J., Libai, B., Muller, E.: Talk of the network: A complex systems look at the underlying process of word-of-mouth. Marketing Letters 12, 211–223 (2001)
5. Gruhl, D., Guha, R., Liben-Nowell, D., Tomkins, A.: Information diffusion through blogspace. SIGKDD Explorations 6, 43–52 (2004)
6. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2003), pp. 137–146 (2003)
7. Kimura, M., Saito, K., Motoda, H.: Blocking links to minimize contamination spread in a social network. ACM Transactions on Knowledge Discovery from Data 3, 9:1–9:23 (2009)
8. Kimura, M., Saito, K., Motoda, H.: Efficient estimation of influence functions fot sis model on social networks. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009 (2009)
9. Kimura, M., Saito, K., Nakano, R.: Extracting influential nodes for information diffusion on a social network. In: Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI 2007), pp. 1371–1376 (2007)
10. Kimura, M., Saito, K., Nakano, R., Motoda, H.: Finding influential nodes in a social network from information diffusion data. In: Proceedings of the 2nd International Workshop on Social Computing, Behavioral Modeling and Prediction (SBP 2009), pp. 138–145 (2009)
11. Kimura, M., Saito, K., Nakano, R., Motoda, H.: Extracting influential nodes on a social network for information diffusion. Data Mining and Knowledge Discovery 20, 70–97 (2010)

12. Kleinberg, J.: The convergence of social and technological networks. Communications of ACM 51(11), 66–72 (2008)
13. Klimt, B., Yang, Y.: The enron corpus: A new dataset for email classification research. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 217–226. Springer, Heidelberg (2004)
14. Leskovec, J., Adamic, L.A., Huberman, B.A.: The dynamics of viral marketing. In: Proceedings of the 7th ACM Conference on Electronic Commerce (EC 2006), pp. 228–237 (2006)
15. Newman, M.E.J.: The structure and function of complex networks. SIAM Review 45, 167–256 (2003)
16. Newman, M.E.J., Forrest, S., Balthrop, J.: Email networks and the spread of computer viruses. Physical Review E 66, 035101 (2002)
17. Richardson, M., Domingos, P.: Mining knowledge-sharing sites for viral marketing. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002), pp. 61–70 (2002)
18. Romero, D.M., Meeder, B., Kleinberg, J.: Differences in the mechanics of information diffusion across topics: Idioms, political hashtags, and complex contagion on twitter. In: Proceedings of the 20th International World Wide Web Conference (WWW 2011), pp. 695–704 (2011)
19. Saito, K., Kimura, M., Motoda, H.: Discovering influential nodes for SIS models in social networks. In: Gama, J., Costa, V.S., Jorge, A.M., Brazdil, P.B. (eds.) DS 2009. LNCS (LNAI), vol. 5808, pp. 302–316. Springer, Heidelberg (2009)
20. Saito, K., Kimura, M., Ohara, K., Motoda, H.: Learning continuous-time information diffusion model for social behavioral data analysis. In: Zhou, Z.-H., Washio, T. (eds.) ACML 2009. LNCS (LNAI), vol. 5828, pp. 322–337. Springer, Heidelberg (2009)
21. Saito, K., Kimura, M., Ohara, K., Motoda, H.: Behavioral analyses of information diffusion models by observed data of social network. In: Chai, S.-K., Salerno, J.J., Mabry, P.L. (eds.) SBP 2010. LNCS, vol. 6007, pp. 149–158. Springer, Heidelberg (2010)
22. Saito, K., Kimura, M., Ohara, K., Motoda, H.: Discovery of super-mediators of information diffusion in social networks. In: Pfahringer, B., Holmes, G., Hoffmann, A. (eds.) DS 2010. LNCS (LNAI), vol. 6332, pp. 144–158. Springer, Heidelberg (2010)
23. Saito, K., Kimura, M., Ohara, K., Motoda, H.: Which targets to contact first to maximize influence over social network. In: Greenberg, A.M., Kennedy, W.G., Bos, N.D. (eds.) SBP 2013. LNCS, vol. 7812, pp. 359–367. Springer, Heidelberg (2013)
24. Sheldon, D., Dilkina, B., Elmachtoub, A., Finseth, R., Sabharwal, A., Conrad, J., Gomes, C., Shmoys, D., Allen, W., Amundsen, O., Vaughan, W.: Maximizing the spread of cascades using network design. In: Proceedings of the Twenty-Sixth Annual Conference on Uncertainty in Artificial Intelligence (UAI 2010), pp. 517–526. AUAI Press, Corvallis (2010)
25. Steeg, G.V., Ghosh, R., Lerman, K.: What stops social epidemics? In: Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM), pp. 377–384 (2011)
26. Watts, D.J.: A simple model of global cascades on random networks. Proceedings of National Academy of Science, USA 99, 5766–5771 (2002)
27. Watts, D.J., Dodds, P.S.: Influence, networks, and public opinion formation. Journal of Consumer Research 34, 441–458 (2007)

SM2D: A Modular Knowledge Discovery Approach Applied to Hydrological Forecasting

Wilfried Segretier and Martine Collard

Mathematics, Computer Science and their Application Laboratory (LAMIA)
University of French West Indies and Guiana (UAG)
{wsegreti,mcollard}@univ-ag.fr
lamia.univ-ag.fr

Abstract. In this paper, we address the problem of flood prediction in complex situations. We present an original solution in order to achieve the main goals of *accuracy*, *flexibility* and *readability*. We propose the SM2D modular data driven approach that provides predictive models for each sub-process of a global hydrological process. We show that this solution improves the predictive accuracy regarding a global approach. The originality of our proposition is threefold: (1) the predictive model is defined as a set of aggregate variables that act as classifiers, (2) an evolutionary technique is implemented to find best juries of such classifiers and (3) the flood process complexity problem is addressed by searching for sub-models on sub-processes identified partly by spatial criteria. The solution has proved to perform well on *flash flood* phenomena of tropical areas known to be hardly predictable. It was indeed successfully applied on a real caribbean river dataset after both preprocessing and preliminary analysis steps presented in the paper.

1 Introduction

Hydrological forecasting systems that predict water levels or trigger flood alerts are mostly designed for specific regional watersheds that share common properties. For instance in Europe, Mediterranean rivers share a typical and well known hydrological behaviour for which alert systems based on physical models have been developed. In this paper, we focus on a particular kind of events called *flash floods* that occur frequently in tropical regions. They are due to massive and very sudden rainfalls. We define the SM2D (Spatial Modular Data Driven) approach to address both efficiency and flexibility issues and to propose a classification method for discrete class variables.

While meteorological models are expected to provide a complement to hydrological models [3] particularly in the case of very sudden rainfalls like so called *flash* events, we show in this paper that observed cumulative rainfalls can be used to identify sub-processes in a way relevant enough to improve the *accuracy* of flash flood prediction in comparison to standard approaches. We meet thus the goal of efficiency. The *flexibility* of the SM2D approach is provided by its modularity: the complex hydrological phenomenon is considered as a collection of sub-processes rather than a unique global one and sub-models are elicited on each sub-process. The concept of sub-process is defined on the basis of various dimensions (hydrometeorological, spatial, data-oriented or knowledge-based). The *readability* of the SM2D approach remains in the kind of predictive models that are elicited by the way of an evolutionary data driven technique previously introduced [8,9].

In addition to accuracy, an important feature for hydrological predictive system is its lead time. It is expected to give the most accurate information about future situations at the earliest time for a public authority to take important decisions for intervention (close roads or open dams for instance). We thus study performances according to two different lead times.

The paper is organized in 7 sections. Section 2 gives an overview of hydrological forecasting systems and particularly data-driven modular techniques. Section 3 presents the source data we used to highlight the approach on the specific case of the *Lezarde* river in the french West Indies, preprocessing operations and preliminary analysis that motivated the design of the modular approach. Section 4 introduces the SM2D approach that we propose and the notion of *aggregate variable* on which relies the search for best predictive readable solutions. Section 5 describes the evolutionary algorithm that we implemented to address the optimization problem resulting from the objective of accuracy. In section 6, we present and discuss the results obtained with the modular approach by comparison with a non modular option of the same technique and other standard machine learning techniques such as decision trees and artificial neural networks. Then we conclude and give some perspectives for future work in Section 7.

2 Related Work

Hydrological forecasting models are often classified according to the way they represent physical processes underlying rivers or water streams activity. *Physically based models* (PBM) typically use differential equations on flows to model the behaviour of streams with respect to multiple parameters such as soil properties, topographical features or evapotranspiration rates. They are also called *spatially distributed models* since they use grid-based representations of river basins. Since they are quite complex and their performances strongly depend on parameters initialization, simpler models have been designed and have proved to perform better in operational and short term flood forecasting [10,3].

Conceptual or *semi-distributed models* are mainly “simplifications” of PBM, they are built on concepts such as the notion of reservoir filling and emptying, on approximations of parameters and on mathematical or empirical knowledge to model the hydrological behaviour of a water stream.

Over the last decades, advanced methods and tools borrowed to the machine learning field have contributed to define a new kind of models, namely *data driven models* (DDM). In the domain of hydrological DDMs, most solutions have used Artificial Neural Networks (ANN), mainly for their ability to model non-linear relationships [11]. But ANNs are black box systems, thus they are not easily interpretable. In most cases, stakeholders are more likely to trust models that express their predictions explicitly. In our work, the long term objective is to design a decision making tool able to provide a comprehensible knowledge representation that explains why a flood or a non-flood is predicted. Thus in this work, the first strong requirement was the *readability* of the models provided to end users that are experts in the hydrological domain.

Modular models (MM) [3,12,13,14] have been proposed to address more precisely the complexity of natural phenomena that they consider as collections of sub-processes rather than global unique processes. The main idea of MMs relies on the notion of *sub-model*: each sub-process identified is modelled by a sub-model and all sub-model outputs are supposed to be combined to produce the global behaviour. Sub-models may be physical, conceptual or data driven. Recent results on MMs tend to show that they improve the prediction accuracy [3]. For instance, global hydrological regression techniques may obtain good average performances over wide time periods but turn to be less accurate during extreme events (drought or massive floods), while flow regimes based MMs, i.e. for which sub-processes are identified according to different flow regimes, can often deal more efficiently with rather different hydrological situations. Solomatine [12] gives a detailed classification of committee machine modular models essentially based on the input space partitioning. In [13], a flow prediction approach is defined in which ANNs and M5 model trees are combined on different sub-processes identified thanks to expert knowledge. In [14], a clustering technique based on Self-Organising Maps (SOM) is used to identify subsets in the input data (assimilated to sub-processes) for which different ANNs are then trained.

In this study, we propose an original modular approach based on a spatial criteria and designed to address the specificity of fast, hardly predictable and devastating flood events that are typical in the caribbean region.

3 Case Studied

3.1 Source Data

The *Lezarde watershed* is the most important hydraulic system of the *Martinique* island in terms of surface area and instrumentation. A critical economical and industrial area is located around the downstream region of the river, making the forecasting of flood events particularly crucial for local authorities. Limnimetric (height water levels) sensors are set up on eight spots - from up to down *Blanche*, *Desirade*, *Gue-Blanche*, *Pompage*, *Pont RN1*, *Scism*, *Soudon*, *Spitz* - along the river course and its two major tributaries. Rain gauges are set up at five locations - from up to down *Bois-Lezard*, *Colson*, *Bois-Parc*, *Olive*, *Cirad* - over the whole surface of the watershed. Figure 1 shows a map of the watershed. The blue lines represents the river course while the green lines shows the watershed borders. Level stations are figured with red squares, while rain gauges are figured with black squares. The up to down length of the area is about thirty kilometers and the distance between two adjacent level stations along the river course is around two to six kilometers. Currently these sensors are monitored by technical services

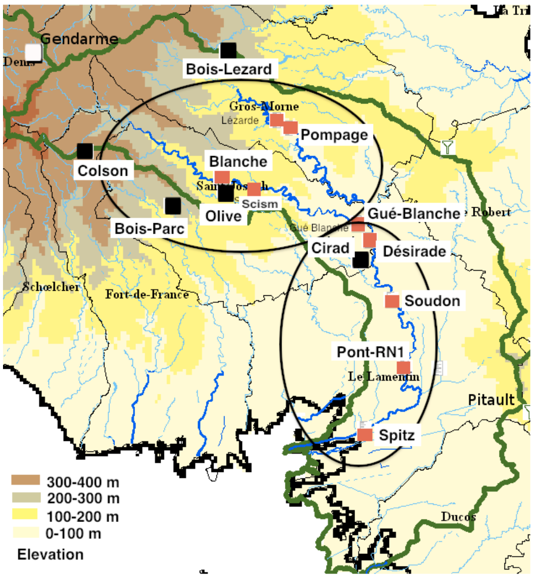


Fig. 1. Map of the *Lezarde* watershed. Level stations are figured with red squares and rain gauges with black squares.

via the regional flood alert system mainly based on alert thresholds. This system involves physical hydrological streamflow models for simulating the river states. According to current end users and decision makers in charge of the public security, this system have two drawbacks: (1) it is not enough explanatory on forecasts and (2), in order to ensure that no error occurs in predicting effective floods, it generates too much false positives.

Source data available for this study consist in thirteen files, each containing the records collected on one sensor during the time period from January 1^{st} 2006 to August 31^{st} 2010. The default record interval, i.e. the default interval between two measures, is six minutes for level stations and one hour for rain stations (cumulative rainfalls). However, missing data corresponding to periods ranging from few hours to several months are quite numerous.

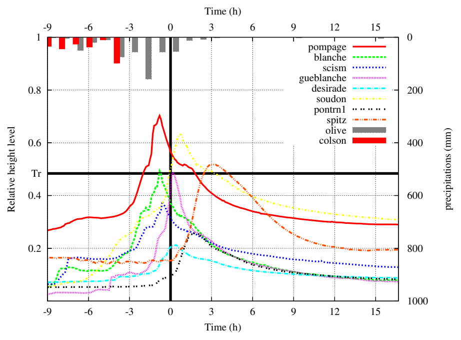
The **preprocessing** step represents a substantial part in a knowledge discovery in databases process since it involves data transformation, normalisation and replacement which are often necessary on noisy or raw source data first to get a better insight in the information they may hide and then to address specific mining tasks. In this study, we encountered two main issues with source data, namely the occurrence of (1) erroneous and (2) missing data. Errors (1) results mostly from technical malfunctions (calibration problems, exposure to natural elements...), thus it happens that the data recorded by a sensor are punctually erroneous. In a first step, we eliminated the most obviously wrong data for each sensor in a semi-automated way thanks to a specific data visualization tool. Missing data (2) are essentially due to the unavailability of sensors during periods up to several months. We found an amount of such missing values ranging from 12.2% to 56.8% depending on the sensor considered. As explained in previous work [9], we defined several replacement strategies depending on the hydrological behavior observed in the spatial and/or temporal neighbourhood of stations. These strategies include linear interpolation, linear regression, non-linear regression (polynomial, ANNs,...) or simply no replacement when not enough information are available on neighbouring stations.

In the following, we call *flood event (F)* (resp. *non-flood event (N)*), a threshold overpassing (resp. non overpassing) recorded by a target sensor at a given time. Thresholds are those defined in the current forecast system for each sensor. We call *event sensor*, the target sensor on which we observe event occurrences and for which we plan to accomplish F or N class prediction and *measure sensor*, a sensor which records are used to predict the event sensor class.

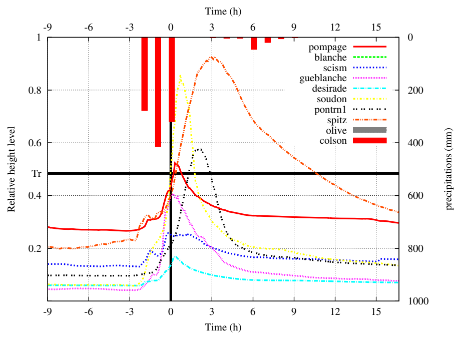
After the preprocessing step, we have searched the whole data set for examples of such events for each class (F and N). Each event is represented by its initial time, i.e. the time when the flood alert threshold is reached for class F or the time when the water level reached a local maximum that stays under the alert threshold for class N. We focused on clear events of type N i.e. when some hydrological activity is observed around the critical area. We chose the sensor *Soudon* as *event sensor* since it is the best tradeoff between the proximity to the critical area and the amount of F and N examples available. We have identified 50 events (21 in class F and 29 in class N), far enough from each other to preserve independence between them. Since there is a time lag between the last value used for prediction and the event time, a same sensor may play both roles of *measure sensor* and *event sensor* at different times. Table 1 details the global structure of the feature space used for all the experiments of this work. Except the first (event number, not used) and the last (class label) columns, the columns corresponds to the values of each level sensor on times $t-prec-m\times s$, $t-prec-(m-1)\times s$, ..., $t-prec$, where L_a^b stand for the value observed on level station b at time a, s is the time step (6 minutes) and m the number of values considered. The class attribute corresponds to the state of the event sensor taken at time t.

Table 1. Global structure of the feature space

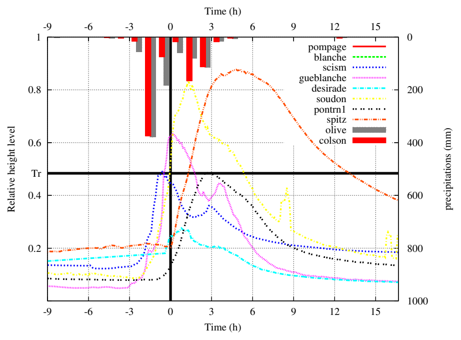
Event	$L_{t-m\times s-prec}^{Station\ 1}$	$L_{t-(m-1)\times s-prec}^{Station\ 1}$	....	$L_{t-prec}^{Station\ 1}$	......	$L_{t-prec}^{Station\ n}$	Class F/N (t)
1	321	?	...	645	......	579	N
2	1235	1267	...	1678	......	2319	F
					....		
					....		
50	1567	1517	...	?	......	3299	F



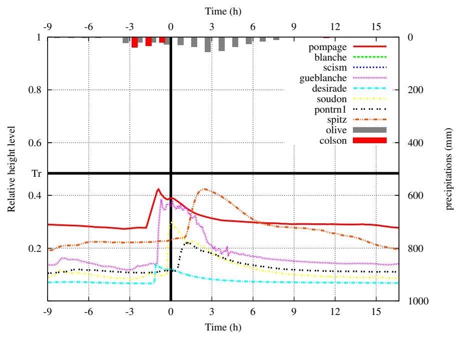
(a) C_{Up} and $\neg C_{Down}$



(b) $\neg C_{Up}$ and C_{Down}



(c) C_{Up} and C_{Down}



(d) $\neg C_{Up}$ and $\neg C_{Down}$

Fig. 2. Four typical hydrological events on *Soudon*, each of one corresponding to instances of the four sub-processes identified for 60 minutes ahead prediction

3.2 Preliminary Analysis

As said before, the hydrological activity observed on the *Lezarde* watershed is a quite fast and complex phenomenon. A visual analysis of the levels and rain stations measurements on available events shows the dynamics of the water activity on this catchment. Figures 2 (a), (b), (c) and (d) show typical flood and non-flood events observed in this watershed for event sensor *Soudon*. The x-axis corresponds to the time in hours, while the left y-axis indicates normalized values of water level and the right up-to-down y-axis indicates hourly cumulative rainfalls in millimeters. The horizontal bolded line represents the flood alert threshold (Tr) defined by domain experts and used in this work to determine flood or non-flood events, while the vertical bolded line shows the time 0, i.e., the beginning of the event. Each colored curve stands for the values of a water level station while each colored histogram bar stands for the values of a rainfall station. We can see on these figures that quite important level rises occur in less than one or two hours and that they are often shortly shifted respectively to the up-to-down order of the stations along the river course, suggesting that the water travel time between stations may be very short (about 20 minutes to more than one hour). We can also observe that, as expected, the rain measurements seem to be good overall predictors of level rises, as they always precede them. However, as underlined in previous studies [9], the variations between the observed relationships from one event to another seem to be highly non-linear and thus not easily explainable. For instance, an important rise on a given station may not be followed on the next station by a proportional rise, the travel time of the water may be different between two given stations, or the relation between observed rain and water levels may change.

4 SM2D Approach

The main idea of MMs is to consider the hydrological process of a watershed as a collection of homogeneous sub-processes each represented by its own sub-model applied for forecasting. To the best of our knowledge, there is no formal definition of the *sub-process* concept in this context, but it appears as something like a part of a generally complex process. A sub-process is sometimes referred as *local* process and conversely a process is then referred as *global*. It is defined on the basis of different conditions on dimensions that allow to select a subset of the global process. Like the latter, it may be represented by a physical, conceptual or data driven model depending on its kind of dimensions.

In order to give a more formal basis to the approach, we have identified four dimensions to define an hydrological sub-process:

– hydro-meteorological (H), i.e., based for instance by conditions defined by water level ranges or flow regime categories,
– spatial (S), i.e., based on geographic portions of the watershed, like sub-basins for instance,
– data-oriented (D), i.e., based on patterns extracted by machine learning techniques such as clustering,
– knowledge based (K), i.e., based on a priori expert domain knowledge, such as seasons characteristics.

In the SM2D approach designed for flash events, sub-processes are defined on the H, S and K dimensions. They can be considered as combinations of specific rainfall behaviour observed on spatial sub-basins. In the following, Section 4.1 presents the modularity side of SM2D and Section 4.2 presents the data-driven solution followed.

4.1 Spatial Modular Approach

A *SM2D sub-process* is defined by a logical expression $C_1\ op_1\ C_2\ op_2\ ...\ op_n C_n$ where each C_i is a logical condition defined on the hydro-meteorological (H), spatial (S) and knowledge-based (K) dimensions and each op_j is a logical operator in $\{and, or, xor\}$. A *SM2D sub-basin* is defined as a geographical portion of the whole studied watershed. A *SM2D sub-model* is a data-driven model designed for a sub-process. For instance, if we consider the sub-basins *Up* and *Down* defined as the upstream and downstream areas of the *Lezarde* watershed and represented roughly by black circles on Figure 1, the logical expression:

Sub-basin Up in active rainfall activity
and
Sub-basin Down in inactive rainfall activity

defines the sub-process $SP1$. As seen in Section 3.2, in the case of tropical short basins that we consider, the impact of localized rainfalls is identified as one of the the factors that dramatically defines the hydrological behaviour of the river. We have indeed considered a priori domain expert knowledge to identify relevant hydrometorological (H) and spatial (S) conditions: the rainfall behaviour and the partitioning of the *Lezarde* watershed into the two *sub-basins* corresponding to its upstream and downstream areas. Each of the two areas is associated to a rainfall station: station *Olive* for sub-basin *Up* and station *Cirad* for sub-basin *Down*.The principle was in this case to identify sub-processes based on rainfall activity and localization. A logical condition C_i within a sub-process represents a rainfall activity measured by a cumulative station. We used thresholds on cumulative rainfall values to characterize the precipitation activity (*active*, *inactive*) in a sub-basin at a given time. If the threshold is reached (resp. not reached) for the station associated to a sub-basin, we consider that the corresponding activity is *active* (resp. *inactive*). These threshold values has been provided by experts on this particular catchment as the knowledge-based (K) dimension of the SM2D approach. Table 2 gives their values in millimeter for both 60 and 120 minutes ahead prediction.

Table 2. Threshold values used on cumulative rainfalls to determine sub-processes

prec (mins)	Olive (mm)	Cirad (mm)
60	40	40
120	35	15

Figures 2 (a), (b), (c) and (d) show typical instances of four sub-processes defined similarly as $SP1$. On Figure 2 (a), we observe a high level of precipitations on *Up* sub-basin (station Olive - yellow bars) only while conversely Figure 2 (b) highlights a high level of precipitations on *Down* sub-basin (station Cirad - green bars) only. Figure 2 (c) shows important rainfalls on both stations while Figure 2 (d) illustrates a sub-process defined by low values on precipitations on both sub-basins. Thus let us assume that:

C_{up} is defined as the condition *Sub-basin Up in active rainfall activity* and C_{down} is defined as the condition *Sub-basin Down in active rainfall activity.* Figures 2 (a), (b), (c) and (d) illustrates the four sub-processes: C_{up} *and* $\neg C_{down}$, $\neg C_{up}$ *and* C_{down}, C_{up} *and* C_{down}, $\neg C_{up}$ *and* $\neg C_{down}$.

Since our objective is to conduct a predictive analysis with classification techniques to predict flood and non flood events, we have identified instances of these sub-processes in situations of flood or non flood event occurring after a given time period called *lead time* or *precocity* following the last rainfall observed measure. Table 3 gives the distribution of events type according to the four sub-processes with a precocity of 60 minutes. We can see that for each sub-process one class is often more represented than the other, suggesting that each sub-process is particularly subject to one kind of events. For instance, as expected, the last situation (d) is highly concerned by non-flood events while (b) or (c) situations are highly concerned by flood events.

One of the common issues in modular approaches is generally to find a way to combine the sub-models discovered in order to generate final prediction. This situations occurs when the input data subsets corresponding to the identified sub-processes overlap. In this approach, we are not concerned by this problem since data subsets are mutually exclusive: a sub-model is trained and tested for each sub-process and directly deployed when validated.

Table 3. Distribution of events type according to each identified sub-process for 60 minutes ahead prediction

	(a) C_{Up} and $\neg C_{Down}$	(b) $\neg C_{Up}$ and C_{Down}	(c) C_{Up} and C_{Down}	(d) $\neg C_{Up}$ and $\neg C_{Down}$
F	3	4	10	4
N	7	0	4	18
Total	10	4	14	22

4.2 Aggregate Variables

The second principle in our approach was to implement the concept of *aggregate variable* that contributes to achieve the goal of *readability*. Sub-models on each sub-process are sets (or juries) of aggregate variables [8,9]. An aggregate variable may be seen as a classifier which predicts the class variable state F or N on the basis of aggregate values of water levels and/or cumulative rainfalls computed over a time period. Its structure is explanatory itself.

More formally, for a given *event* sensor *es*, an aggregate variable (AV) is defined by a tuple $(ms, F, prec, agg, thresh)$ of parameters which value ranges are customized for the given watershed and where *ms* is a *measure sensor* which records are considered for aggregation, F is an *aggregation function* such as mean, standard deviation, minumum etc. , *prec* is a *precocity period* corresponding to the time interval between the last measure considered and the initial time of the event on *es* to predict, *agg* is an *aggregation period* over which the aggregate variable value is computed and *thresh* is a set of thresholds defined on F values range and used to determine output predictions. Figure 3 provides a temporal representation of an AV applied for prediction. For instance, if the *event sensor* is *Soudon*, AV=(*Gue-Blanche*, average, 132, 96, $\{500\}$) represents the average value computed on *Gue-Blanche* levels recorded during 96 minutes and with 132 minutes precocity before an event on *Soudon*. If this computed value is greater than 500, then a flood event is predicted, else a non-flood event is predicted.

The *spatial modular approach* proposed in this work consists in learning *predictive sub-models* on sub-processes as juries of such variables. Juries of AVs can be seen as sets of AVs combined in such a way that the final decision of the jury corresponds to the majority vote of its AVs. When a precocity p_{jury} is fixed for a jury, it means that the precocity of its AVs can be superior or equal to p_{jury}. If an AV is not able to take a decision at a given time because of the unavailabilty of its measure sensor data, two strategies can be used: the AV is discarded thus the majority vote is computed on the remaining AVs or a decision is taken for the AV, for instance arbitrarily or on the basis of neighbouring stations behaviour.

Since each of the parameters in each AV of a jury can take numerous values (depending on the number of sensors available, the granularity of earliness and aggregation periods or the variation ranges of hydrological values), the search for optimal juries of aggregate variables turns to a combinatorial optimization problem which search space consists of all the possible instances of variables sets, i.e., $|S| = \prod_{i=0}^{n} |S_i|$ with $|S_i| = \prod_{j=0}^{k} |V_j|$ the search space of the i^{th} AV, $|V_j|$ the size of the value set for the j^{th} parameter and n the number of AVs of a jury. Obviously, $|S|$ grows quite rapidly with n and $|V_j|$, becoming impractical. Thus, to address this problem, we have implemented an evolutionary algorithm as a stochastic method.

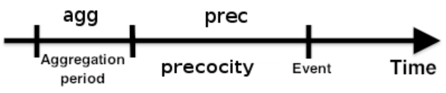


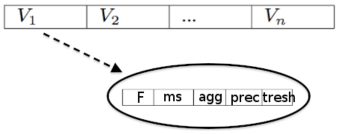
Fig. 3. Schematic temporal representation of an AV

5 Evolutionary Approach

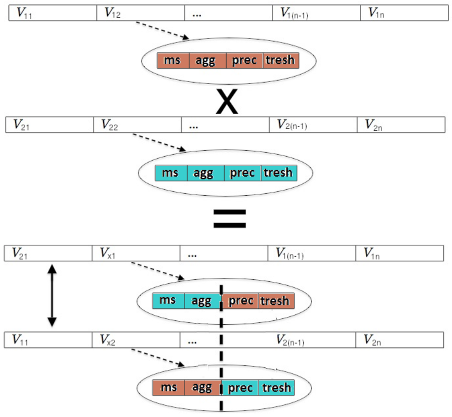
Evolutionary algorithms (EA) [4,1] are global search heuristics inspired by the main principles of the neo-darwinian theory of evolution. A population of solutions (individuals) undergoes evolution by iteratively applying stochastic operators such as selection, recombination (crossover) and variation (mutation). These methods have proved to be efficient to find useful solutions for optimization and search problems with acceptable execution time comparatively to exhaustive techniques. In this section, we describe the evolutionary algorithm that we developed to address the problem of searching the best sets of AVs for prediction.

5.1 Individual Representation and Genetic Operators

An *individual* is a representation of a jury of AVs. One AV is represented by the non ordered set of its parameters. It is a simple classifier that can predict an event sensor class. As seen before, a set of AVs is also a classifier, its answers are



(a) Individual representation



(b) Crossover scheme

Fig. 4. Evolutionary algorithm features

defined as a combination of its AVs answers by a majority vote. The number of AVs within a set is a constant. Figure 4 (a) shows the structure of an individual made of n AVs $V_1, V_2, ..., V_n$.

The *crossover* is a genetic operator which main purpose is to bring novelty into the population by allowing individuals - the parents - to recombine in order to generate new individuals - the offsprings - made of their parents genetic heritage. It is generally seen as an exploitation operator since it takes advantage of the existing population. It can be defined on one or several points. We have defined a multi-point crossover operator consisting in the one point recombination of each variable of the set, i.e., each variable of an individual is recombined with another variable of the other individual at a randomly defined point. Figure 4 (b) illustrates this operator. In an EA process, the role of the *mutation* operator is mainly to prevent the convergence in local optima by allowing to explore rather different regions of the search space. It is generally applied at very low rates that favour the exploitation. We have defined a uniform mutation operator consisting simply in changing the value of one parameter randomly chosen in an individual by another possible value.

5.2 Objective Function

The essential role of the **objective function** is to guide the evolutionary process by assigning a score to each individual of the population evaluating their quality regarding the problem to solve.

In this context of binary classification (F and N events) in which good true positive (TP) and true negative (TN) rates should be obtained with a jury of n AVs, we have considered the *Matthews Correlation Coefficient* [5] as the objective function:

$MCC = \frac{(TP \times TN)-(FP \times FN)}{\sqrt{(TP+FP)\times(TP+FN)\times(TN+FP)\times(TN+FN)}}$ with TP, TN, FP, FN respectively the number of true positives, true negatives, false positives and false negatives. It is widely used in machine learning to evaluate the performances of a binary classifier. Its major advantage regarding other methods is to be useful even with high class imbalance like in this work. Its value set is $[-1;1]$: 1 for perfect classification, 0 for not better than random classification, and -1 for perfect inverse classification, i.e, all TPs are classified as TN and vice versa. In a first time we also considered the well known $F_\beta Measure$ based on precision and recall. However, unlike MCC, its major drawback is that it does not take into account TN and thus can be seen as a "positive evaluator" carrying more about TP. In this work, two values can be considered for the evaluation of a jury performances: first the individual performance of each AV in the jury and then the performances of the jury itself. We combined these two indicators in the following weighted sum to compute the fitness value of a jury $F_{jury} = a \times MCC_{jury} + b \times \sum_{i=1}^{n} MCC_i$, where MCC_i the MCC of the i^{th} AV and $a, b > 0$.

In order to obtain the experimental results presented in the next section, we defined the following parameters for the EA: 8000 generations, a population size of 100, crossover and mutation rates of 0.75 and 0.01 respectively, a stochastic binary tournament selection and an elitist replacement (i.e., the best individual of the precedent generation is kept if needed). The *stochastic tournament selection* results in a low selection pressure avoiding premature convergence and is a low computational cost schema since it selects the best individual among two that are uniformly drawn from the population. We also used a Random Offspring Generation (ROG) [7] strategy in our crossover so that when to identical individuals are selected to be crossed, the crossover operators is not applied since it would be useless, but two offspring are returned, one being and exact copy of its two parents while the other is randomly generated. This technique allows to introduce diversity into the evolutionary process with the advantage of not disturbing it as it is typically the case when the mutation rate is simply increased to high values.

6 Experimental Results

In this section we present and discuss the results obtained with the SM2D approach applied on data recorded on the *Lezarde* watershed and we compare them to (1) the same evolutionary search for optimal aggregate variables as a global approach, and to (2) machine learning techniques commonly used in data driven hydrological modelling either as (a) traditional global approaches or as (b) modular approaches according to the sub-processes identified in Section 4.1.

As said previously in Section 3, the sensor *Soudon* has been set as the *event sensor*, we have found 50 events (21 in class F and 29 in class N) and the four typical sub-processes C_{up} *and* $\neg C_{down}$, $\neg C_{up}$ *and* C_{down}, C_{up} *and* C_{down}, $\neg C_{up}$ *and* $\neg C_{down}$ have been identified. Each of these sub-processes has been associated to a subset of the total input data available as explained in Table 3. However, since the subsets corresponding to sub-processes C_{up} *and* $\neg C_{down}$ and $\neg C_{up}$ *and* C_{down} were too small in comparison to the others, we decided to merge them in a C_{up} *xor* C_{down} sub-process defined by the exclusive disjunction of rainfall activity observed either on the upstream or downstream area.

Moreover, since the size of each subset was quite small in comparison to the full dataset, we applied a *leave one out* cross validation (loocv, i.e. k-fold cross validation with k set as the total number of samples minus one) technique in order to estimate the performances of each model. However, since this size issue was less significant for global approaches, we used a typical stratified 10-fold cross validation technique in this case. An important point to note is that the performances showed for each k-fold cross validation experiment represents the average performances of the k models learnt.

Since evolutionary algorithms and some standard techniques (Random-Forest) that we used are stochastic, i.e. based on random properties (as opposed to deterministic techniques such as C4.5), each of their associated experiment has been averaged on 100 runs in order to have statistically significant results. The standard deviation is given in order to provide a confidence coefficient on each of these results[1].

In the following, we present experiments conducted with two precocity periods: 60 and 120 minutes. As shown in Figure 1, the input values for each model consists of all the available measures of each station during the five hour before the begining of the precocity time, i.e., $m = 50$.

Table 4 gives the main parameters that we used for each standard technique. An important parameter of our approach is the size of the juries searched, i.e., the number of aggregate variables. Obviously, the more numerous are the aggregate variables in a jury, the more accurate can be the fit to the problem addressed. However the main drawback of this situation is that the size of the search space, and thus the complexity of the search process, increases dramatically with the jury size. As we showed in [9], for this particular flood classification problem,

Table 4. Parameters used for each standard technique

Technique	Parameters
MLP	# hidden layers: 1, # nodes:$(\lvert attributes\rvert + \lvert classes\rvert)/2 = 202$ learning rate: 0.3, momentum 0.2
C4.5	minimum instances per leaf: 2, pruning: yes confidence factor for pruning: 0.25
RF	maximum depth: unlimited,# of tree generated:10 # of feature used in random selection:$log(\lvert attributes + 1\rvert)$

[1] 10-fold cross validation experiments with deterministic techniques have also been averaged on 100 experiments since the repartition of samples into folds is done randomly. However the averaging was not necessary for leave one out cross validation experiments since in this particular case the repartition into folds is deterministic thus no standard deviation is given in this case.

Table 5. Performances of sub-models on sub-processes for standard techniques and SM2D

prec (mins)	Technique		C_{Up} *and* C_{Down}			C_{Up} *xor* C_{Down}			$\neg C_{Up}$ *and* $\neg C_{Down}$		
			TN	TP	W.Avg	TN	TP	W.Avg	TN	TP	W.Avg
60	C4.5	Avg	50.00	80.00	71.40	57.10	71.40	64.30	94.40	50.00	86.40
		Std Dev	-	-	-	-	-	-	-	-	-
	RF	Avg	56.25	89.00	79.64	78.69	76.98	77.86	98.10	33.25	86.33
		Std Dev	18.49	3.00	5.74	13.64	9.67	8.67	2.65	17.70	3.99
	MLP	Avg	48.00	83.00	72.99	85.70	56.24	70.97	93.74	50.00	85.85
		Std Dev	6.78	4.58	3.87	0.00	3.37	1.68	1.79	0.00	1.49
	SM2D	Avg	68.00	68.40	68.28	99.42	62.28	90.85	99.88	72.50	94.90
		Std Dev	15.03	6.74	7.44	2.80	6.10	3.21	0.77	7.50	1.48
120	C4.5	Avg	50.00	0.00	25.00	90.00	80.00	86.70	76.90	60.00	69.60
		Std Dev	-	-	-	-	-	-	-	-	-
	RF	Avg	63.35	46.66	54.99	91.20	32.80	71.73	90.45	62.80	78.45
		Std Dev	9.43	16.33	10.26	8.16	17.32	7.25	4.50	9.60	5.06
	MLP	Avg	66.70	45.99	56.31	68.60	33.20	56.78	66.12	47.80	58.18
		Std Dev	0.00	7.13	3.54	4.90	9.47	3.84	5.55	6.41	4.25
	SM2D	Avg	66.66	58.33	62.50	99.60	71.20	90.13	94.92	62.20	80.69
		Std Dev	11.05	18.33	10.03	1.96	9.93	3.59	3.95	10.06	4.93

very reduced sets of variables can obtain better or equivalent performances than higher ones with the advantage of requiring less computational power. Thus all the results that we present in the following has been obtained with juries of 3 aggregate variables.

Table 5 shows a comparison between the performances obtained by three commonly used data driven techniques, namely C4.5 [6], Random Forest (RF) [2], a Multi-Layer Perceptron (MLP) (Artifial Neural Network), and the proposed SM2D approach applied on the data corresponding to each sub-process separately. The first column gives the precocity time, the second column gives the technique used while next columns gives the true positive rate (TP, i.e, percentage of flood samples correctly classified as flood), true negative rate (TN, i.e., percentage of non-flood samples correctly classified as non-flood) and weighted average (W.Avg, i.e., $\frac{|F| \times TP + |N| \times TN}{|F|+|N|}$) performances for each sub-process. We can see that for both 60 and 120 *prec* time, the best W.Avg performances for the sub-processes C_{up} *xor* C_{down} and $\neg C_{up}$ *and* $\neg C_{down}$ have been provided by aggregate variables (90.85 and 94.90 for 60 mins, and 90.13 and 80.69 for 120 mins). The best performances are also provided by AVs with 120 mins of precocity for the C_{up} and C_{down} sub-process (62.50). Table 6 shows a comparison between the global performances and the averaged modular approach performances, i.e. a weighted average of the performances of each sub-model, for each standard technique used. We can observe that for 60 minutes prediction, the modular approach with aggregate variables is better either than each global approach or other modular approach with common data driven techniques. For 120

Table 6. Comparison of global and modular approaches performances for standard techniques and SM2D

prec (mins)	Technique		Global			W.Avg Mod.		
			TN	TP	W.Avg	TN	TP	W.Avg
60	C4.5	Avg	78.81	71.19	75.60	79.27	71.41	76.01
		Std Dev	3.82	5.57	3.06	-	-	-
	RF	Avg	92.09	63.19	79.94	87.64	74.37	82.08
		Std Dev	3.18	4.68	2.69	7.49	8.02	5.79
	MLP	Avg	80.43	60.71	72.14	85.49	67.79	78.08
		Std Dev	4.73	5.20	2.94	2.70	3.30	2.20
	SM2D	Avg	97.93	61.33	82.56	95.37	67.14	86.31
		Std Dev	1.82	4.22	1.96	3.22	6.67	3.63
120	C4.5	Avg	79.74	58.98	71.44	75.85	47.62	64.16
		Std Dev	4.89	3.51	2.98	-	-	-
	RF	Avg	86.46	66.43	78.04	85.10	51.04	70.80
		Std Dev	3.10	6.08	3.03	6.78	13.36	6.96
	MLP	Avg	88.23	54.46	74.04	67.09	43.80	57.31
		Std Dev	4.76	4.52	3.38	4.18	7.34	3.96
	SM2D	Avg	98.41	62.76	83.44	90.46	63.23	79.15
		Std Dev	2.09	3.89	1.96	4.73	12.39	5.75

minutes prediction, the modular approach with aggregate variables is still better than other DDM techniques (either global or modular). We observe however that it seems under the global approach with AVs.

These results are quite promising and tend to confirm our first assumption that flash flood event classification can be addressed effectively by both aggregate variables and an hydrometeorological based modular approach, particularly for very short term prediction (60 minutes). However, since the number of events available for this study was quite small, these results should be confirmed on more numerous data.

7 Conclusion

In this paper we proposed a knowledge discovery approach relying on a typical data mining flow made of pre-processing, classification, data aggregation and a stochastic method to achieve accurate forecasts. Data from two types of measures - water levels and cumulative rainfalls - were considered. We proposed an original rainfall based modular approach in order to address the complexity of the hydrological process leading eventually to disastrous flash flood phenomena that are not well modelized particularly in the caribbean region. With this proposition, we aim at providing an alternative to standard hydrological systems that do not perform well on very fast and massive phenomena known as *flash foods.* In the field of data-driven hydrological modelling systems, this approach is quite original. We have obtained first encouraging results and our further works will consist in scaling them up, for instance by using more numerous data or other performance metrics such as the area under ROC curve analysis. We also plan to investigate fuzzy approaches for the modelling of sub-processes which boundaries are not always crisp.

References

1. Back, T., Fogel, D.B., Michalewicz, Z. (eds.): Handbook of Evolutionary Computation. IOP Publishing Ltd., Bristol (1997)
2. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
3. Corzo, P., Perez, G.: Hybrid Models for Hydrological Forecasting: Integration of Data-Driven and Conceptual Modelling Techniques. Taylor & Francis Group (2009)
4. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press (1975)
5. Matthews, B.W.: Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochimica et Biophysica Acta 405(2), 442–451 (1975)
6. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
7. Rocha, M., Neves, J.: Preventing premature convergence to local optima in genetic algorithms via random offspring generation. In: Imam, I., Kodratoff, Y., El-Dessouki, A., Ali, M. (eds.) IEA/AIE 1999. LNCS (LNAI), vol. 1611, pp. 127–136. Springer, Heidelberg (1999)
8. Segretier, W., Clergue, M., Collard, M., Izquierdo, L.: An evolutionary data mining approach on hydrological data with classifier juries. In: IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE (2012)
9. Segretier, W., Collard, M., Clergue, M.: Evolutionary predictive modelling for flash floods. In: IEEE Congress on Evolutionary Computation (to appear, 2013)
10. Sene, K.: Hydrometeorology: Forecasting and Applications. Springer (2010)
11. Solomatine, D.P., Price, R.K.: Innovative approaches to flood forecasting using data driven and hybrid modelling. In: Proceedings of the 6th International Conference on Hydroinformatics, pp. 1–8 (2004)
12. Solomatine, D.P., Siek, M.B.: Modular learning models in forecasting natural phenomena. Neural Networks: The Official Journal of the International Neural Network Society 19(2), 215–224 (2006)
13. Solomatine, D.P., Xue, Y.: M5 Model Trees and Neural Networks: Application to Flood Forecasting in the Upper Reach of the Huai River in China 9(6), 491–501 (2004)
14. Toth, E.: Classification of hydro-meteorological conditions and multiple artificial neural networks for streamflow forecasting. Hydrology and Earth System Sciences 13(9), 1555–1566 (2009)

A Dynamic Programming Algorithm for Learning Chain Event Graphs

Tomi Silander and Tze-Yun Leong

National University of Singapore

Abstract. Chain event graphs are a model family particularly suited for asymmetric causal discrete domains. This paper describes a dynamic programming algorithm for exact learning of chain event graphs from multivariate data. While the exact algorithm is slow, it allows reasonably fast approximations and provides clues for implementing more scalable heuristic algorithms.

Keywords: chain event graphs, structure learning, model selection.

1 Introduction

Chain event graphs (CEGs) [1] have recently gained popularity for representing asymmetric causal domains in which some statistical independences hold in some contexts but not in others [2]. Evaluating the structures of these models based on data has been briefly addressed earlier by Freeman and Smith [3]. However, to our knowledge, the only work on actually learning these models from data appears in the 2010 PhD thesis of Guy Freeman [4]. In this work he proposes and demonstrates an agglomerative hierarchical clustering heuristic (AHC) for learning maximum a posterior (MAP) CEG structures from data.

The AHC algorithm leaves much to hope for as discussed in Freeman's thesis. The algorithm relies on the fixed variable ordering which is not always available. Even for a fixed ordering AHC does not scale well with the size of data.

In this paper we propose adapting the dynamic programming approaches used for exact structure learning of Bayesian networks [5,6] for learning structures of chain event graphs. The resulting method learns CEG structures without requiring predetermined ordering of the variables. When approximated with a fast clustering heuristic, the algorithm can also scale up to about 30 variables and we will demonstrate the algorithm using 21 different data sets. We also suggest using factorized normalized maximum likelihood [7] as a scoring function since this avoids the problem of Bayesian structure learning being very sensitive to Dirichlet priors [8,9].

We will next introduce chain event graphs in a form that is useful for explaining our structure learning algorithm. We will then discuss the decomposability properties of structure evaluation criteria that are required for our algorithm to work. In Sect. 4 we will first represent the exact structure learning algorithm for a fixed variable ordering, and then generalize this to the case in which the ordering of variables is not known beforehand. Before concluding, we will discuss more scalable approximations of the exact algorithm and demonstrate their practicality with an experiment on real data sets.

2 Chain Event Graphs

Chain event graphs are a model family for multivariate discrete data. CEGs encode regularities in probability trees [10], and due to their heavy reliance on variable ordering, they are claimed to be particularly suitable for representing causal domains [2]. They allow representing asymmetric independence relations where independence statements are conditioned on particular configurations of a subset of random variables [1] unlike in Bayesian networks in which independences must hold given any configuration of some random variables [11]. In the following, we will present a formulation for so called n-homogeneous chain event graphs [1] that are a natural choice when learning CEGs from data without much prior knowledge.

2.1 Data

The data used for learning chain event graphs consists of N independent instantiations of an n-dimensional random vector $V = (V_1, \ldots, V_n)$, where each discrete random variable V_i takes (without loss of generality) one of the values in $dom(V_i) = \{1, \ldots, r_i\}$. We will be using vectors and their prefixes extensively, so we denote the i^{th} component of the vector X by X_i and the vector consisting of the first i coordinates of the vector X by $X_{:i}$. For example, using this notation we have $dom(V_{:i}) = \prod_{l=1}^{i} dom(V_l)$. We assume the data to contain no missing values.

2.2 Model

The following introduction to CEG models aims at allowing us to express the likelihood function for the data which is later needed to present the structure learning problem as an optimization problem. For a more general introduction, one may consult a paper by Smith and Anderson [1]. We will first describe the graphical structure of CEGs as networks with nodes and arcs, and then populate the structure with parameters that turn the structure into a statistical model that defines the likelihood of a data vector.

CEG Structure. To represent a CEG we have to first fix the (total) order of variables. A CEG structure G for n-dimensional data is a layered directed acyclic graph with $n + 1$ layers (see Fig. 1). The first n layers of G correspond to the variables in the given ordering. The last layer has only one node and it does not correspond to any variable. To keep the mathematical notation simple, we assume that the variables have been renamed (reordered) so that the layer $i \in \{1, \dots, n\}$ in CEG corresponds to the variable V_i in the data. We will separately consider different orderings later in Sect. 5. Each layer $i \leq n$ consists of nodes (often called *positions* in the CEG literature) and from each node there leave exactly r_i arcs each of which points to a node in the next layer $i+1$. The r_i arcs from any node at layer i are all labelled by different values of V_i. All the nodes except the single first one at layer 1 must have at least one arc pointing to them. We say that arcs starting from the nodes at layer i belong to the layer i.

Nodes of CEG Layer as Data Partitions. By following consecutive labels, a CEG structure defines a unique path from the start node at layer 1 to the end node at layer $n+1$ for any given complete data vector d. We notice that the nodes at layer $i > 1$ partition the domain $dom(V_{:i-1})$, i.e., for each node t at layer i we can assign the set $dom(t) = \{d_{:i-1} \mid d \in dom(V) :\ d \ traverses\ t\}$ of possible data vector prefixes that lead to the node t. Similarly, the arcs at layer i partition the domain $dom(V_{:i})$, i.e., the arc $a = (t, v)$ from node t labeled by v can be associated with the set $dom(a) = \{d_{:i} \mid d \in dom(V) :\ d \ traverses\ a\}$ of data vector prefixes. We denote the unique node in which the vector prefix $d_{:i-1}$ leads to by $t(d_{:i-1})$, and the unique last arc in the path of $d_{:i}$ by $a(d_{:i})$.

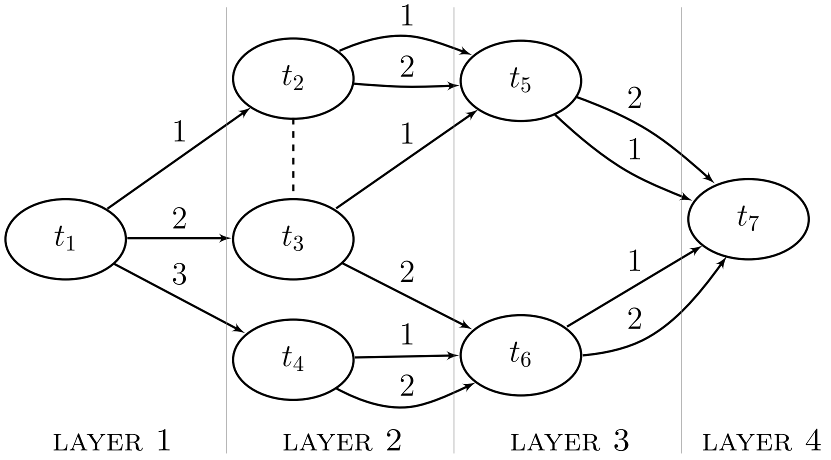


Fig. 1. A CEG structure for three variable domain V in which V_1 has three values and the other variables two. The nodes t_2 and t_3 belong to the same stage, while all the other nodes form stages of their own. The path of data vector $(2, 1, 2)$ traverses nodes t_1, t_3, t_5, t_7, and for node t_6 at layer 3 we have $dom(t_6) = \{(2, 2), (3, 1), (3, 2)\}$.

CEG as a Statistical Model. A CEG structure can be turned into a multivariate probability distribution by assigning discrete probability distributions to the nodes. More specifically, let V_i be any variable and t a node at layer i. We then attach a conditional probability defining parameter $\theta_a = \theta_{t,v} = P(V_i = v|V_{:i-1} \in dom(t))$ to the arc leaving the node t that is labeled by the value v. Since each data vector d defines a unique path in CEG, it is easy to see (by applying the chain rule) that multiplying the parameters (conditional probabilities) on the path of d defines a probability for the data vector d. Assuming that the data vectors in D are generated independently from the CEG model gives the likelihood function for the data D as a product of likelihoods of individual data vectors.

Introducing Stages for Parameter Sharing. While the model described above does indeed define a valid joint probability distribution, CEGs try to capture even more regularity by allowing different nodes at the same layer to share distributions thus reducing the number of parameters in the model. To formalize this parameter sharing, the nodes in one layer are partitioned into *stages* and each stage is assigned a single distribution, thus each node in a stage has the same distribution. Notice that nothing in the likelihood function really changes, but in order to be a valid CEG model, the conditional probability distributions assigned to the nodes of the same stage must be equal.

Since a node corresponds to the set of paths that lead to the node, the stages can be associated with sets of these sets. We denote the stage to which the node t belongs as $s(t) = \{dom(u)|u \sim t\}$, where the node partition induced equivalence relation '$\sim$' denotes "belonging to the same stage". We also define a shorthand notation $s(d_{:i}) \equiv s(t(d_{:i}))$ for the stage the data-vector prefix $d_{:i}$ leads into. Accommodating the stages, we will now denote the parameter for the arc labelled v that starts from a node t by $\theta_{s(t),v}$. We denote the set of stages partitioning the nodes of layer i by L_i, and write the likelihood function as

$$P(D|G,\theta) = \prod_{d\in D}\prod_{i=1}^{n} \theta_{s(d_{:i-1}),d_i} = \prod_{i=1}^{n}\prod_{s\in L_i}\prod_{v=1}^{r_i} \theta_{s,v}^{N_{s,v}} \ , \tag{1}$$

where $N_{s,v}$ denotes the total number of data vectors in D that traverse any node in stage s and any arc labeled with value v of V_i. Indeed, the frequencies $N_{s,v}$ form the sufficient statistics for the CEG G, and the histograms $\boldsymbol{N_s} = (N_{s,1}, \dots, N_{s,r_i})$ in layer L_i are the key entities involved in learning the CEG structures.

From this formulation it is easy to see that the maximum likelihood parameters are just the relative frequencies of data vectors that traverse the arcs, i.e., in layer i we have

$$\hat{\theta}_{s,v} = \frac{N_{s,v}}{\sum_{v'=1}^{r_i} N_{s,v'}} = \frac{N_{s,v}}{N_s} \ , \tag{2}$$

whenever the denominator N_s is non-zero. In case none of the data vectors traverse the stage s, the maximum likelihood parameters are not identifiable, and we may choose to set them to (say) r_i^{-1}. The actual maximal likelihood for the data is naturally independent of this choice.

Introduction of stages separates the node sequence based conditioning structure from the stage sequence based structure that defines the likelihood function. This important property will be later exploited in our exact structure learning algorithm.

3 Scoring CEG Structures

Probabilistic models aim at being good summaries of the data or the data generating process. Measuring the extent to which a model succeeds in this task for a particular data is often done by using some evaluation criterion or scoring function. The structure learning task is then cast as an optimization problem in which we try to find the highest scoring structure G from the structure space for the given data D. We will next review some of the popular scoring functions and present their forms for the CEG structures.

The algorithms in Sects. 4 and 5 rely on the additive structure of the scoring function that the algorithm maximizes. The scoring function SC should decompose the same way the likelihood function does, i.e., it should be a sum of scores SC_s for individual stages s:

$$SC(G;D) = \sum_{i=1}^{n} SC_{L_i}(G;D) = \sum_{i=1}^{n} \sum_{s \in L_i} SC_s(G;D) \ . \tag{3}$$

It is easy to see that the common model selection criteria such as the Bayesian information criterion (BIC) [12], the Akaike information criterion (AIC) [13], the Bayesian Dirichlet score (BD) [3], and the factorized normalized maximum likelihood (fNML) score [7] decompose conveniently this way. For all penalized maximum likelihood scores, the stage scores in layer i can be expressed in a common form $SC_s(G;D) = \sum_{v=1}^{r_i} N_{s,v} \log \hat{\theta}_{s,v} - C$, where the complexity terms C for different scores are $C_{AIC} = r_i - 1$, $C_{BIC} = \frac{r_i - 1}{2} \log N$, and $C_{fNML} = \log \sum_{D'_i} P(D'_i | \hat{\theta}(D'_i))$, where D'_i goes over all the possible datasets of size N_s for a discrete variable having r_i values. The terms C_{fNML} can be quickly computed with methods developed by Kontkanen et al. [14].

For the BD score, the stage score may be expressed as

$$SC_s(G;D) = \log \frac{\Gamma(\sum_{v=1}^{r_i} \alpha_{s,v})}{\Gamma(\sum_{v=1}^{r_i} \alpha_{s,v} + N_{s,v})} + \sum_{v=1}^{r_i} \log \frac{\Gamma(\alpha_{s,v} + N_{s,v})}{\Gamma(\alpha_{s,v})} \ , \tag{4}$$

where $\boldsymbol{\alpha}_s = (\alpha_{s,1}, \ldots, \alpha_{s,r_i})$ are the Dirichlet distribution hyperparameters for independent parameters $\boldsymbol{\theta}_s = (\theta_{s,1} \ldots, \theta_{s,r_i})$ [3]. Since specifying informative parameter priors for large CEG structures is often tedious, it is customary, especially for data mining purposes, to set the prior by assuming uniform "pseudo-data", i.e., that we had seen all the possible data vectors N' times. Even under this simple prior the MAP structure is likely to be very sensitive to the choice of N' [3,9]. Silander et al. suggest avoiding this problem by using the fNML score [7].

The scores above are designed to be used for observational data. Analogous decomposable scores for a mixture of experimental and observational data may also be derived as demonstrated by Cooper and Yoo [15].

4 Algorithm for Fixed Variable Order

We will now present an algorithm that learns a globally optimal CEG structure for the given decomposable score and the data D when the order of variables is fixed. For simplicity, we again assume that this order matches the order of variables in the data. Somewhat surprisingly, it turns out that the layers of a CEG structure can be optimized independently without taking into account the structure of the other layers, since the two consecutive individually optimized layers can always be linked together without affecting the score of the graph.

4.1 Optimal Partitions for Layers

As discussed earlier, for the decomposable scoring functions considered in this paper, the layer score SC_{L_i} depends on the goodness of its stages. The first layer always consists of a single node thus it is necessarily optimal. The stages of the layer $L_i, 1 < i \leq n+1$ define a unique partitioning of the domain $dom(V_{:i-1})$. In reverse, for all i, $(1 < i \leq n)$, any partition of the domain $dom(V_{:i-1}) = \bigcup_{s=1}^{m} S_s$, $(x \neq y \Rightarrow S_x \cap S_y = \emptyset)$, suggests a layer L_i which can be scored since it defines values $N_{s,v} = \sum_{d \in D} I(d_{:i-1} \in S_s \wedge d_i = v)$, where I is the indicator function. The partition into m parts will introduce $m(r_i - 1)$ parameters and each part S_s can also be associated with a r_i-dimensional vector $\boldsymbol{\alpha}_s$ of Dirichlet hyperparameters.

Since domain $dom(V_{:i-1})$ is finite, it has a finite number of possible partitions. (The number of these partitions is called the $|dom(V_{:i-1})|^{th}$ Bell-number [16].) Since all the partitions can be scored using the data D, we suggest a procedure $bestLayer(V_i, V_{:i}, D)$ that goes through all the partitions of $dom(V_{:i-1})$ and returns one with a maximal score. The optimal partition is seldom unique, but it is trivial to see that the layer score S_{L_i} of any valid CEG structure cannot exceed the highest partition score of $dom(V_{:i-1})$. The sequence of Bell-numbers grows very rapidly, thus the proposed brute-force algorithm is not practical; we will address this issue later in Sect. 6.

For future development, we notice that the values $N_{s,v}$ in layer i count the frequencies of certain variable assignments to variables $\{V_1, \ldots, V_i\}$ in data D. The number of these assignments does not really depend on the order of variables, thus instead of partitioning the subsequence space $dom(V_{i-1})$, we might as well partition the space of possible assignments to the variables $\{V_1, \ldots, V_{i-1}\}$. Therefore, we may take the second argument to the $BestLayer$-procedure to be a set of variables rather than a sequence of them.

Algorithm 1. Compacting layers

```
Require: an array of layers $L$
  # one layer is a set of stages
  # one stage is a set of nodes
  # one node is a set of data vector prefixes

  # Reserve space for compacted layers
  $L_{new} \leftarrow$ array of $|L|$ empty sets
  $L_{new}[|L|] \leftarrow \{\{dom(V)\}\}$

  # Compact stages of each layer starting from the last layer
  for $i \leftarrow |L| - 1$ downto 1 do
      for $S \in L[i]$ do
          $nodes \leftarrow$ an empty hashtable from layer $i+1$ node-vectors to prefix sets
          for $\{pfx\} \in S$ do
              $targets \leftarrow (t_{new}(pfx+1), t_{new}(pfx+2), \ldots, t_{new}(pfx+r_i))$
              if $targets \notin keys(nodes)$ then
                  $nodes[targets] \leftarrow \emptyset$
              end if
              $nodes[targets] \leftarrow nodes[targets] \cup \{pfx\}$
          end for
          $S_{new} \leftarrow \{nodes[k] \mid k \in keys(nodes)\}$
          $L_{new}[i] \leftarrow L_{new}[i] \cup \{S_{new}\}$
      end for
  end for

  return $L_{new}$
```

4.2 Binding Layers Together

We will next argue that the optimal partitions of domains $dom(V_{:i-1})$ can be turned into optimal layers L_i of a valid CEG structure. We do that by building the layer L_{n+1} first and then working backwards all the way down to L_1.

The last layer is easy to build since it always has just one stage with one node t_{end} with $dom(t_{end}) = dom(V)$. Working backwards to build a layer L_i, $(i \leq n)$, we may now assume that the nodes of layer L_{i+1} have already been built. We then go through parts S_s in an optimal partition of $dom(V_{:i-1})$, and turn each S_s into a stage with a separate node for each member of the set S_s. For each member $d_{:i-1}$ of $dom(V_{:i-1})$ and for all values v of V_i, we then draw an arc labelled with v from node $t(d_{:i-1})$ to the node $t(d_{i:-1}, v)$ in the next layer. The construction guarantees that the result is a valid CEG structure and that its score equals the sum of scores of optimal partitions for separate layers.

4.3 Compacting CEG

The construction above yields an unwieldy CEG structure with layer L_i having $|dom(V_{:i-1})|$ nodes. Undeniably, this is somewhat disappointing development for the quest for learning a parsimonious model for the data. We therefore suggest a cleaning phase that is carried out for each layer L_i as soon as it is constructed. We first notice that we may merge those nodes that belong to the same stage and whose similarly labeled arcs point to same nodes in the next layer. For example, in the layer L_n this rule yields just one node per stage, and due to this, the layer L_{n-1} is also likely to be considerably pruned, and so on. The pseudocode for compacting the stages of the maximal CEG is presented in Alg. 1.

Searching for a compact representation of the optimal CEG structure is complicated by the existence of nodes and arcs that have no support in data. For AIC and BIC the optimal partitions do not contain parts without any data support since such parts do not affect the likelihood but they add parameters to the model. However, unsupported nodes in the layer may be arbitrarily merged and distributed into stages without affecting the score which creates a huge source of non-identifiability. The fNML score is not affected by unsupported parts in the partition, but again the unsupported nodes may be arbitrarily merged and distributed to the stages. For the BD score the situation is actually more difficult since the score affecting hyperparameters $\boldsymbol{\alpha}_s$ are attached to the stages that may contain data prefixes not occurring in the data D. We find this an additional reason to prefer the fNML score over BD score.

A possible solution to the non-identifiability dilemma could be to merge all the unsupported nodes of a layer L_i into a "gutter" node and place that into its own stage that is associated with a fixed uniform-distribution for V_i. This way the number of free parameters in the model does not change, and the score of the model, for other than the BD score, would still be optimal. Such a gutter gathers the unsupported data vectors, and the resulting model still defines the probability of any possible data vector after the free parameters have been estimated (or marginalized out).

5 General CEG Structure Learning Algorithm

This far we have assumed that the variable ordering is given to us beforehand. While this may sometimes be the case in causal and/or temporal domains, in general we do not necessarily have such prior information. In some cases, finding a good ordering of variables may be the very problem we are trying to solve. Some variable orderings may indeed allow higher scoring models than other orderings.

5.1 Dynamic Programming

The most naive algorithm would go through all the $n!$ variable orderings of the variable set V and apply the algorithm for fixed order learning to each of them. However, one quickly realizes that, due to the decomposability of scores, common prefixes of such orderings need to be studied only once. This suggests a solution very similar to the dynamic programming solutions for the exact learning of Bayesian network structures [5,6]. The score of any globally optimal CEG structure G^* can be expressed as a sum of the last layer's score and the

Algorithm 2. Exact learning of an optimal CEG

```
Require: Variable set $V$, data $D$
Require: functions $layerScore$, and $bestLayer$

  # Reserve space for arrays
  $ls \leftarrow$ array for at most $|V|$ best last layer scores
  $lastVar \leftarrow$ empty table of $2^{|V|}$ variables
  $gs \leftarrow$ empty table of $2^{|V|}$ global scores

  # Find last variables of best CEGs for all variable subsets
  $gs[\emptyset] = 0$
  for all non-empty $U \subset V$ in order of increasing size do
      for all $V_i \in U$ do
          $ls[V_i] \leftarrow layerScore(bestLayer(V_i, U, D))$
      end for
      $lastVar[U] \leftarrow \arg\max_{V_i \in U}(ls[V_i] + gs[U \setminus \{V_i\}])$
      $gs[U] \leftarrow ls[lastVar[U]] + gs[U \setminus \{lastVar[U]\}]$
  end for

  # Gather layers in reverse order
  $L^* \leftarrow$ initially empty array of $n$ layers
  while $V \neq \emptyset$ do
      $L^*[|V|] \leftarrow bestLayer(lastVar[V], V, D)$
      $V \leftarrow V \setminus \{lastVar[V]\}$
  end while

  return $L^*$
```

score of the preceding $n-1$ layers. Consequently, if we have computed the best CEG structures for all variable subsets of size $n-1$ and the last layer scores for all the n variables, we can choose the last layer (variable) that maximises the sum of scores of the last layer and the score of the best CEG for the rest of the variables. This reasoning can then be recursively applied for finding the best CEG for any set of $n-1$ variables.

When reversing the recursion described above into a dynamic programming algorithm, we may first find the best CEGs for all the variable subsets of size 1, then try to augment these with the last layers to find the best CEGs for all the variable subsets of size 2, then try to augment these with last layers again, and so on, until we reach the subset of size n. The pseudo-code for this strategy is presented in Alg. 2. The algorithm takes the variable set V and the data D as an input, and it also uses the functions *bestLayer* and *layerScore* to find and score the best last layer in CEG for variable set S in which the last variable is V_i.

5.2 Time and Space Complexity

We notice that the function *bestLayer* and the summation under *arg max* are called once for each possible $(V_i, S \setminus \{V_i\})$ combination thus $n2^{n-1}$ times.

This determines the computational complexity of this level of the algorithm. The total complexity is naturally affected by the possibly huge complexity of the *bestLayer*-search.

The required space for storing last variables for subsets to the table *lastVar* is clearly $2^n - 1$ (bytes), but in implementation these may be written to the disk as soon as they are computed. Assuming a deterministic scheme for going over the variable subsets, n of the last variables can then be later fetched from disk when gathering layers into the sequence L^* of optimal layers. Storing the

scores of best CEGs for all subsets of V to gs clearly requires storing 2^n floating point values. We notice that when going through subsets in order of increasing size, only the scores for subsets of two consecutive sizes are required to be held in memory. This reduces the maximum number of floating point values we need to keep in memory to $\binom{n}{\lfloor 0.5n \rfloor} + \binom{n}{\lfloor 0.5n \rfloor + 1} = \frac{2n!}{\lfloor 0.5n \rfloor!(\lfloor 0.5n \rfloor + 1)!}$.

6 Scalable Approximations

Bell-numbers grow super-exponentially. In order to find a maximally good partitioning of the 6^{th} layer for binary data, the brute first algorithm has to go through $B_{32} = 128064670049908713818925644$ different partitions and score them all. While the calculations can be done in parallel (for all layers) this is clearly prohibitive. The problem is slightly alleviated by noticing that, for other scores than BD, only the partitioning of the data supported part of the domains matters, thus in effect we are actually trying to find the good partition of the set of observed data prefixes $D_{:i-1}$. Still, having 32 different data prefixes in our data would lead to studying all B_{32} possible partitions of them.

6.1 Avoiding Brute Force Search

To avoid studying all the partitions of the layer, Freeman suggests starting from the finest clustering of the data prefixes and then using agglomerative clustering that greedily combines the partitions that yield frequency histograms that lead to the best improvement to the score of the layer [4]. In our experiments (see Table 1) this AHC (agglomerative hierarchical clustering) method produced good results, but due to its cubic time complexity, it was rather slow so we were not able to use it for all of our experiments.

In order to devise a faster clustering heuristic, we first note that the quality of the partition is related to the entropy of the distributions entailed by the maximum-likelihood (or expected) parameters $\hat{\boldsymbol{\theta}}_f$ of the stages and the amount of support the parameters have in the data. Ideally one would like to have a small number of low-entropy distributions with strong support.

Prefixes $d_{:i-1}$ in $dom(V_{:i-1})$ that occur in data induce frequency histograms $\boldsymbol{N}_{s(d_{:i-1})}$. Clustering prefixes with similar shape low-entropy histograms together tend to create partitions of data that yield high scores. In our experiments we used K-means clustering of normalized histograms with Euclidean distance initializing the cluster centers to low-entropy distributions. Figure 2 shows how the residual sum of squares, the measure optimized by K-means, correlates with the scores of the last layer partitions in the Symptoms-data used by Freeman and Smith [3] to demonstrate the CEG learning. This heuristic is hardly optimal, but it is relatively fast (see Table 1).

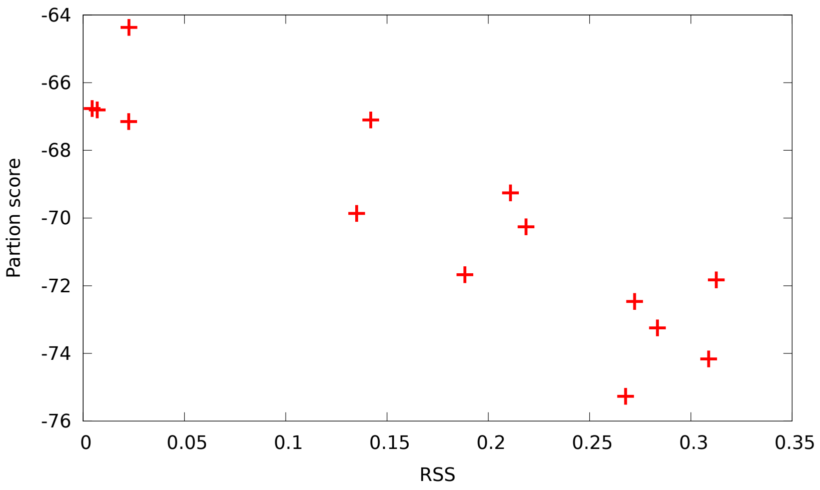


Fig. 2. Partition scores as a function of the K-means objective function

One could also first use a quick clustering heuristic with a modified Alg. 2 that learns k best orderings of variables as suggested by Tian et al. [17], and then evaluate these orderings using more accurate but slower layer optimizing algorithm like AHC.

6.2 Tapping into Bayesian Network Learning

A possible method to quicken the learning is to just consider partitions induced by some short sub-sequences of $V_{:i}$. This corresponds to selecting "parents" among predecessors in a K2-like fashion [18] which connects the CEG learning directly to the Bayesian network structure learning (with local structure) [19]. Doing so also imports a well developed pack of tricks for Bayesian network structure learning such as the ability to quickly learn CEG-"trees" by maximum spanning tree algorithms [20], caching sufficient statistics into the AD trees for quick retrieval [21] etc. However, it also biases search towards domains excluding conditioning contexts that consist of particular configurations of many variables. This bias is undesirable since catching the variable length contexts is the bravura of CEGs. A more CEG-spirited way would be to constrain the maximum width of the layers by limiting the maximum number of parts in a sub-domain partition. For example, in our K-means implementation we have allowed the layer for variable V_i to be partitioned into at most to $2^{r_i} - 1$ parts.

Table 1. Comparison of CEG structure learning and BN structure learning using fNML score

Data	n	N	S_B	S_{CK}	S_{CA}	T_B	T_{CK}	T_{CA}
symptom	3	100	-202.9	-200.2	-200.2	0.00	0.00	0.0
balance	5	625	-4478.4	-4086.7	-4086.7	0.00	0.00	2.3
iris	5	150	-450.9	-440.4	-440.2	0.00	0.00	0.0
thyroid	6	215	-572.4	-559.4	-557.8	0.00	0.00	0.0
liver	7	345	-1299.4	-1231.8	-1227.8	0.00	0.00	0.1
ecoli	8	336	-1643.6	-1508.4	-1496.3	0.00	0.01	0.4
diabetes	9	768	-3654.0	-3447.9	-3393.0	0.02	0.02	2.5
post operative	9	90	-639.9	-481.4	-478.7	0.00	0.02	0.7
yeast	9	1484	-7849.0	-7488.7	-7407.7	0.02	0.20	10.8
breast cancer	10	286	-2739.3	-1897.6	-1864.7	0.04	0.42	95.1
tic tac toe	10	958	-9162.4	-6871.5	-6840.0	0.08	0.26	1203.6
bc wisconsin	11	699	-3239.6	-2507.0	-2489.1	0.07	0.21	100.4
glass	11	214	-1233.2	-1016.7	-1004.5	0.04	0.11	5.7
heart cleveland	14	303	-3352.3	-1988.2	-1969.2	0.98	3.83	3122.2
heart hungarian	14	294	-2343.6	-1787.1	-1766.5	0.56	1.68	630.5
heart statlog	14	270	-2814.3	-1733.7	-1714.1	0.82	2.74	1887.2
wine	14	178	-1808.7	-1148.2	-1140.0	0.62	1.81	713.3
australian	15	690	-4953.0	-4296.0	-4196.1	1.30	4.03	2420.7
credit screening	16	690	-6035.7	-4756.1	-4641.7	5.27	89.78	36883.6
voting	17	435	-3122.8	-2423.5	-2377.8	5.38	10.45	7622.2
primary tumor	18	339	-3074.9	-2105.4	-2056.4	13.77	32.29	25007.2

6.3 An Experiment

It is possible to transform any Bayesian network structure G_B to a CEG structure G_C with exactly the same value of decomposable scoring function. Consequently, a possible "CEG-learning" algorithm would be to actually learn a Bayesian network, and then convert that to a CEG. To demonstrate that the Alg. 2 actually beats trivial turning of a Bayesian network into a CEG, we conducted an experiment in which we took 20 UCI datasets[1] [22] and the Symptoms data by Freeman et al. [3], and used the fNML score to first learn a best Bayesian network for each dataset using exact structure learning, and then learned CEG structures using the Alg. 2. The results are collected to the Table 1 that also lists the number N of data vectors and the number n of variables for all datasets. In all datasets our CEG-learning algorithm was able to find a higher scoring CEG structure than the one converted from the highest scoring (S_B) Bayesian network. Using the AHC algorithm as a *BestLayer*-search yielded the highest score (S_{CA}), but since it is very slow (T_{CA}, measured in seconds)[2] for higher

[1] Continuous values were discretized into 4 equal with bins and the missing values were randomly imputed.

[2] All the experiments were run using a single 3.4Ghz CPU of a Dell Optiplex 990 desktop computer with 8GB of memory (a small fraction of which was actually needed).

dimensional datasets, we also included the faster (T_{C^K}) K-means heuristic described above. While K-means as a *BestLayer*-search never yielded better results than AHC, its score (S_{C^K}) was always better than the one given by the Bayesian network.

We take these results as a demonstration of the usefulness of our algorithm. However, the complexity still heavily depends on dimensions of the data set. Even after distributing the Alg. 2 with the K-means heuristic to 16 processors, it took almost 5 days to run it for the 28-variable Wisconsin diagnostic breast cancer data that has 569 cases.

7 Future Work

Equipped now with a practical algorithm to learn CEGs, we conducted a 10-fold cross-validation study in which we first learned the Bayesian network and CEG structures, calculated the factorized sequential normalized maximum likelihood (fsNML) parameters [7] for these models, and then observed the log-probability of the test-fold in these two models. For each data the cross-validation was repeated 100 times with different foldings. The amount of model learning required made us to use the K-means heuristic as a layer optimizer.

This predictive task forces us to decide what to do with parameters that have no support in the training data. In Bayesian networks, the unseen parent configurations usually yield uniform distributions for the children. For CEGs we therefore implemented a "gutter" as discussed earlier. If a path defined by a test vector turns unsupported, all the remaining variables are predicted with uniform distributions.

As we can see from the results in Table 2, even if the CEG models achieved higher training scores than Bayesian networks (train S_{C^K} vs. train S_B with standard deviations), their ability to predict future UCI-data was worse than Bayesian networks' (test S_{C^K} vs. test S_B). However, for the Symptoms-data [2] that features strong contextual independences that cannot be captured by any Bayesian network structure, the optimal CEGs predict test data better than the optimal BNs do.

We cautiously hypothesize that the "marginal likelihood" is not enough for predictive model selection in this way nested model families. CEG's ability to fit the structure into particular value configurations allows higher likelihoods, but it also seemingly yields wider "gutter", i.e., the number of variables that has to be predicted with uniform distribution increases. This is also reflected with very high standard deviations of the CEG test scores. For example, the huge standard deviation in the credit screening data is mostly caused by an attribute that has 14 different discrete values which causes the test data to contain many variable configurations that do not appear in the training data.

It is worth emphasizing that the cross-validation study above was mainly conducted to demonstrate that our algorithm is practical in a way that it can be used to investigate interesting questions such as model selection of CEGs. The actual problem of model selection remains an interesting topic for future research.

Table 2. Results from 100 10-fold cross-validations

Data	n	N	train S_B	train S_{CK}	test S_B	test S_{CK}
symptom	3	100	-1829 ± 2	-1810 ± 3	-202±5	-196 ± 5
balance	5	625	-40373 ± 3	-36792 ± 98	-4413±14	-6348 ± 730
iris	5	150	-4093 ± 1	-3996 ± 7	-419±11	-429 ± 21
thyroid	6	215	-5186 ± 5	-5066 ± 5	-543±16	-559 ± 67
liver	7	345	-11718 ± 9	-11122 ± 20	-1290±38	-1372 ± 126
ecoli	8	336	-14869 ± 7	-13587 ± 29	-1578±29	-1681 ± 268
diabetes	9	768	-32940 ± 17	-30953 ± 114	-3625±66	-3868 ± 442
post operative	9	90	-5762 ± 13	-4312 ± 8	-664±65	-871 ± 107
yeast	9	1484	-70783 ± 25	-67364 ± 178	-7720±56	-8021 ± 788
breast cancer	10	286	-24710 ± 13	-16971 ± 90	-2705±82	-3909 ± 832
tic tac toe	10	958	-82626 ± 158	-61137 ± 50	-9032±367	-13611 ± 1705
bc wisconsin	11	699	-29208 ± 44	-22460 ± 43	-3250±204	-3979 ± 1928
glass	11	214	-11186 ± 18	-9185 ± 31	-1176±124	-1325 ± 332
heart cleveland	14	303	-30243 ± 26	-17781 ± 44	-3340±223	-4714 ± 960
heart hungarian	14	294	-21145 ± 15	-15988 ± 39	-2323±78	-3291 ± 1158
heart statlog	14	270	-25396 ± 24	-15465 ± 34	-2806±222	-4004 ± 890
wine	14	178	-16372 ± 33	-10289 ± 17	-1769±200	-2613 ± 665
australian	15	690	-44648 ± 25	-38440 ± 113	-4949±122	-6302 ± 7727
credit screening	16	690	-54470 ± 59	-42516 ± 165	-5958±171	-9681 ± 18879
voting	17	435	-28123 ± 55	-21649 ± 43	-3254±363	-4048 ± 894
primary tumor	18	339	-27701 ± 72	-18742 ± 51	-3202±405	-4244 ± 1493

8 Conclusion

We have presented an exact structure learning algorithm for the CEG model family, and suggested approximations that make it applicable in practice. The algorithm learns CEG structures without requiring the predetermined ordering of variables. The approximations still have much room for study and improvement. The exponential nature of even the approximate version of the algorithm ensures that this will not be the general solution for larger CEG structure learning problems. We, however, hope that the work will inspire the community to build on these ideas to come up with better algorithms with possible approximation bounds as well as more general heuristics that scale up to a larger number of variables. We feel that results of data clustering research may well provide ideas to come up with good algorithms for the CEG structure learning in the future.

Acknowledments. This research is supported by Academic Research Grants MOE2010-T2-2-071 and T1251RES1211 from the Ministry of Education in Singapore.

References

1. Smith, J.Q., Anderson, P.E.: Conditional independence and chain event graphs. Artificial Intelligence 172(1), 42–68 (2008)
2. Thwaites, P., Smith, J.Q., Riccomagno, E.: Causal analysis with chain event graphs. Artificial Intelligence 174(12-13), 889–909 (2010)
3. Freeman, G., Smith, J.Q.: Bayesian MAP model selection of chain event graphs. Journal of Multivariate Analysis 102(7), 1152–1165 (2011)
4. Freeman, G.: Learning and predicting with chain event graphs. PhD thesis, University of Warwick (2010)
5. Ott, S., Miyano, S.: Finding optimal gene networks using biological constraints. Genome Informatics 14, 124–133 (2003)
6. Singh, A., Moore, A.: Finding optimal Bayesian networks by dynamic programming. Technical report, Carnegie Mellon University (June 2005)
7. Silander, T., Roos, T., Myllymäki, P.: Learning locally minimax optimal Bayesian networks. International Journal of Approximate Reasoning 51(5), 544–557 (2010)
8. Steck, H., Jaakkola, T.S.: On the Dirichlet prior and Bayesian regularization. In: Advances in Neural Information Processing Systems 15, Vancouver, Canada, pp. 697–704. MIT Press (2002)
9. Silander, T., Kontkanen, P., Myllymäki, P.: On sensitivity of the MAP Bayesian network structure to the equivalent sample size parameter. In: Parr, R., van der Gaag, L. (eds.) Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI 2007), pp. 360–367. AUAI Press (2007)
10. Shafer, G.: The Art of Causal Conjecture. MIT Press, Cambridge (1996)
11. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers, San Mateo (1988)
12. Schwarz, G.: Estimating the dimension of a model. Annals of Statistics 6, 461–464 (1978)
13. Akaike, H.: Information theory and an extension of the maximum likelihood principle. In: Petrox, B., Caski, F. (eds.) Proceedings of the Second International Symposium on Information Theory, Budapest, pp. 267–281. Akademiai Kiado (1973)
14. Kontkanen, P., Myllymäki, P.: A linear-time algorithm for computing the multinomial stochastic complexity. Information Processing Letters 103(6), 227–233 (2007)
15. Cooper, G., Yoo, C.: Causal discovery from a mixture of experimental and observational data. In: Proceedings of the Fifteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI 1999), pp. 116–125. Morgan Kaufmann, San Francisco (1999)
16. Sloane, N.J.A.: The On-Line Encyclopedia of Integer Sequences. Bell or exponential numbers: ways of placing n labeled balls into n indistinguishable boxes, `http://oeis.org/A000110A000110`
17. Tian, J., He, R., Ram, L.: Bayesian model averaging using the k-best Bayesian network structures. In: Proceedings of the Twenty-Sixth Annual Conference on Uncertainty in Artificial Intelligence (UAI 2010), Corvallis, Oregon, pp. 589–597. AUAI Press (2010)
18. Cooper, G., Herskovits, E.: A Bayesian method for the induction of probabilistic networks from data. Machine Learning 9, 309–347 (1992)

19. Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D.: Context-specific independence in Bayesian networks. In: Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence, UAI 1996, pp. 115–123. Morgan Kaufmann Publishers, San Francisco (1996)
20. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Transactions in Information Theory 14(3), 462–467 (1968)
21. Moore, A., Lee, M.S.: Cached sufficient statistics for efficient machine learning with large datasets. Journal of Artificial Intelligence Research 8, 67–91 (1998)
22. Frank, A., Asuncion, A.: UCI machine learning repository (2010)

Mining Interesting Patterns in Multi-relational Data with N-ary Relationships

Eirini Spyropoulou[1], Tijl De Bie[1], and Mario Boley[2]

[1] Intelligent Systems Lab, University of Bristol, UK
[2] Fraunhofer, IAIS, Germany

Abstract. We present a novel method for mining local patterns from multi-relational data in which relationships can be of any arity. More specifically, we define a new pattern syntax for such data, develop an efficient algorithm for mining it, and define a suitable interestingness measure that is able to take into account prior information of the data miner. Our approach is a strict generalisation of prior work on multi-relational data in which relationships were restricted to be binary, as well as of prior work on local pattern mining from a single n-ary relationship. Remarkably, despite being more general our algorithm is comparably fast or faster than the state-of-the-art in these less general problem settings.

1 Introduction

Pattern mining research has mostly focused on mining itemsets or association rules on one binary table representing transactions and items. Real world data is often more complex, commonly stored in a relational database that allows to capture the relations that exist among the different entity types in the data.

Mining multi-relational data has for a long time been approached by Inductive Logic Programming (ILP) methods where the goal is to mine frequent rules about the data [5,12]. More recently, there have also been generalisations of itemset mining to more complex settings.

Itemset mining has been generalised to mining local patterns in a (single) relationship that involves several entity types. Such a relationship is known as *n-ary relationship* [3,8,9]. An example of a 3-ary relationship is a shop transaction record containing the item sold, the customer, as well as the mode of payment used (e.g. cash/debit card/credit card). The definition of a closed itemset can be generalised fairly directly to n-ary relationships. Algorithms designed to mine such patterns typically require choosing a frequency threshold for each entity type, and do not define an interestingness to rank the patterns found.

Another strand of research has generalised itemset mining to *multi-relational data*. Think for example of a retail database which contains customers, items they bought as well as characteristics of customers and characteristics of items. The most common strategy has been to essentially apply frequent itemset mining on the join of all tables of the database [11,7,10]. Although this strategy seems sensible, in previous work [13,14] we argued that it has significant disadvantages.

We thus proposed the so-called Maximal Complete Connected Subset (MCCS) pattern syntax as an alternative, which is a natural generalisation of closed itemsets and their supporting transactions. We also proposed a global measure of subjective interestingness. While we were able to demonstrate promising results, a restriction is that the method can be applied to multi-relational data only if all relationships in the database are binary.

Contributions in this paper. Although both research strands have remained largely independent, real data is often multi-relational while also involving relationships of arity larger than 2. To the best of our knowledge no local pattern mining method exists targeting this case, and we aim to fill this gap in the current paper. In doing this, we wanted to ensure backward compatibility of the newly introduced notions with both these strands of research, ideally without sacrificing computational tractability. More specifically, we introduce a novel pattern syntax called N-MCCSs and show that it is a generalisation of pattern syntaxes defined for simpler settings; namely of MCCSs for multi-relational data with binary relationships [13,14], as well as of n-Sets for data represented by a single n-ary relationship [3].

Developing an algorithm to mine N-MCCSs proved non-trivial owing to the differences in algorithmic approaches for both strands of research. Nonetheless, our experiments show that the proposed algorithm performs similarly to the algorithm for mining MCCSs from multi-relational data with binary relationships only [14], and on n-ary data it considerably outperforms Data-Peeler [3], the state-of-the-art algorithm for mining data containing a single n-ary relationship. Thus, our contributions simultaneously *generalize and unify important previous data mining methods* and *approximately match or even improve on them in terms of computation times.*

2 Definitions and Concepts

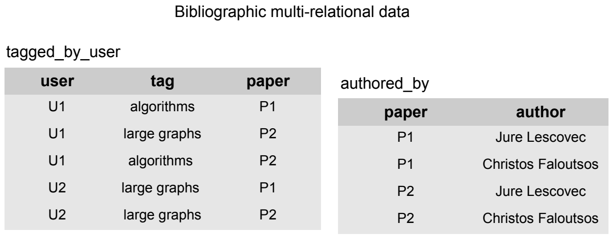
Here we formalize multi-relational data as considered in this paper and define the concepts needed to introduce the pattern syntax we propose.

2.1 Multi-relational Data

Our formalization of multi-relational data follows the Entity-Relationship (ER) model [6], with the difference that we treat every attribute as an entity type of its own. This unified treatment of entity types and attributes leads to a simple formalization which still captures all the associations in the data.

We formalize **multi-relational data** as $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$. E is a finite set of **entities** which is partitioned into k **entity types**. We formalize this by defining a mapping from the entity set onto an index set for the types: $\mathbf{t} : E \to T = \{1 \ldots, k\}$. Thus, the set of entities is the union of the sets of entities for each type: $E = E_1 \dot{\cup} \ldots \dot{\cup} E_k$ with $E_i = \{e \in E \mid \mathbf{t}(e) = i\}$. Slightly abusively, we will also apply $\mathbf{t}$ to sets of entities, to yield a set with their entity types.

We also define a **relationship type** R as a set of entity types, such that the set of all relationship types $\mathbf{R}$ is a subset of the power set of T, i.e.: $\mathbf{R} \subseteq \mathcal{P}(T)$. For each $R \in \mathbf{R}$ there is an $|R|$-ary relationship $\mathcal{R}_R \subseteq \{\{e_1, \ldots, e_n\} : R = \{\mathbf{t}(e_1), \ldots, \mathbf{t}(e_n)\}\}$, containing all sets of entities of the types in R that are indeed related with each other. We say that a set $\{e_1, \ldots, e_n\} \in \mathcal{R}_R$ is a **relationship instance**. The set $\mathcal{R}$ is the union of all relationships, i.e., $\mathcal{R} = \bigcup_{R \in \mathbf{R}} \mathcal{R}_R$. Figure 1 shows a toy example of some multi-relational data highlighting the notions we just introduced.


Bibliographic multi-relational data

tagged_by_user

user	tag	paper
U1	algorithms	P1
U1	large graphs	P2
U1	algorithms	P2
U2	large graphs	P1
U2	large graphs	P2

authored_by

paper	author
P1	Jure Lescovec
P1	Christos Faloutsos
P2	Jure Lescovec
P2	Christos Faloutsos

Fig. 1. Bibliographic multi-relational data containing four entity types ("user", "tag", "paper", "author") and two relationship types. One 3-ary between "user", "tag" and "paper" and one binary between "paper" and "author".

2.2 The Pattern Syntax of N-MCCSs

The pattern syntax proposed in this paper is called N-MCCS and builds upon two previously considered ones: the pattern syntax of MCCSs, which was proposed for the case of multi-relational data with many binary relationships and is a generalisation of closed itemsets or maximal tiles [13,14], and the pattern syntax of n-Sets, which was proposed for the case of data with one n-ary relationship only [3]. In this section we define N-MCCSs and in the next section we show that it is indeed a generalisation of MCCSs and n-Sets.

Intuitively, N-MCCSs capture associations between entities of different entity types. In the example of Fig. 1, an N-MCCS could correspond to a group of users that have tagged the same group of papers of the same authors or a group of users that have tagged the same group of papers with the same tags.

The definition of N-MCCSs is based on the notions of *connectedness* and *completeness*. A set of entities is connected if any pair of entities in it is connected through a sequence of entities that are pairwise related. Formally:

Definition 1. *We call a set of entities $F \subseteq E$ **connected** if for all distinct $e, e' \in F$ there is a sequence $e = e_1 \ldots e_l = e'$ with $\{e_1..e_l\} \subseteq F$ such that for $i \in \{1, \ldots, l-1\}$ it holds that there is an $F' \supseteq \{e_i, e_{i+1}\}$ such that $F' \in \mathcal{R}$.*

Example 1. In the toy data example of Fig. 1 the set of entities {U1, U2, large graphs, P2, Christos Faloutsos} is connected whereas the set of entities {U1, U2, large graphs, Christos Faloutsos} is not connected because for none of the pairs {U1, Christos Faloutsos}, {U2, Christos Faloutsos}, {large graphs, Christos Faloutsos} is there a superset which is a subset of any relationship.

A set of entities is complete if every subset containing entities of different entity types that belong to the same relationship type, is a subset of a relationship instance. One might expect that for completeness we would require all entity types of a relationship type to be present in a pattern. However our definition of completeness is more flexible. In the example of Fig. 1 for instance it allows patterns containing only a set of papers and their tags without containing the users. We formally define completeness using the notion of a critical subset.

Definition 2. *Given some relational data $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$, and a set of entities $F \subseteq E$, a set $F' \subseteq F$, with $|\boldsymbol{t}(F')| = |F'|$ and $\boldsymbol{t}(F') \subseteq R \in \boldsymbol{R}$ is called a* ***critical subset*** *of F with respect to R. We say that a critical subset, F' of F,* ***is covered*** *in $\mathcal{R}$ if for all $R \in \boldsymbol{R}$ with respect to which it is critical, there exists an $F'' \supseteq F'$ such that $F'' \in \mathcal{R}_R$.*

Definition 3. *Given some relational data $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$, we call a set of entities $F \subseteq E$* ***complete*** *iff all critical subsets of F are covered in $\mathcal{R}$.*

Example 2. Considering again the data toy example of Fig. 1 the set{U1, U2, large graphs, Christos Faloutsos} is complete but not connected.

A set of entities $F \subseteq E$ is called a Complete Connected Subset and abbreviated as **N-CCS** if it satisfies both completeness and connectedness as defined here for multi-relational data of n-ary relationships. Like in other pattern mining tasks the set of N-CCSs is typically exponentially larger than the input size. For reasons of efficiency of the mining algorithm, we focus on a smaller subset of the N-CCSs namely the maximal N-CCSs. A maximal N-CCS, abbreviated as **N-MCCS**, is an N-CCS to which no entity can be added without violating connectedness or completeness.

Example 3. Two examples of N-MCCSs from the toy data of Fig. 1 are: {U1, U2, large graphs, P2, Christos Faloutsos, Jure Lescovec} and {P1, P2, large graphs, algorithms, Christos Faloutsos, Jure Lescovec}.

2.3 From MCCSs and n-Sets to N-MCCSs

Here we show that the proposed pattern syntax of N-MCCSs is a generalisation of MCCSs [13,14], as well as n-Sets [3]. We do this by translating the definition of MCCSs and n-Sets using the concepts of this paper.

MCCSs are Maximal Complete Connected Subsets where completeness and connectedness are defined as follows. For $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$ where $\forall R \in \mathbf{R}$, $|R| = 2$, a set $F \subseteq E$ is complete if $\forall F' \subseteq F$ with $t(F') \in \mathbf{R}$ and $|F'| = 2$ it holds that $F' \in \mathcal{R}_{t(F')}$. A set $F \subseteq E$ is connected if for all $e, e' \in F$ there is a sequence $e = e_1, \ldots, e_l = e'$ with $\{e_1, \ldots, e_l\} \subseteq F$ such that for $i \in \{1, \ldots, l-1\}$ it holds that $\{e_i, e_{i+1}\} \in \mathcal{R}$. Clearly both completeness and connectedness are special cases of completeness and connectedness as defined in Def. 3 and Def. 2 respectively.

n-Sets are defined as follows. For some multi-relational data $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$ where $|\mathbf{R}| = 1$ and for $R \in \mathbf{R}$, $|R| = n$, a set $F \subseteq E$ with $|\mathbf{t}(F)| = n$ is an

n-Set if $\forall F' \subseteq F$ with $|\mathbf{t}(F')| = |F'| = n$, $F' \in \mathcal{R}_R$ and it is maximal under this condition. Since $|\mathbf{R}| = 1$, and for every n-Set F, $|\mathbf{t}(F)| = n$, if F is complete it will be connected as well. Therefore the pattern syntax of N-MCCSs is a generalisation of the pattern syntax of n-Sets.

3 Mining Algorithm

In Sec. 2.2 we argued that for efficiency reasons we focus on mining maximal patterns. An efficient algorithm to mine all N-MCCSs should avoid enumerating most N-CCSs that are not maximal. The algorithm proposed here, is an instantiation of the fixpoint listing algorithmic framework [2] which is used to enumerate all N-CCSs that are fixpoints of a closure operator and to which we refer as *closed N-CCSs*. In Sec. 3.1 we establish the applicability of this framework for mining N-MCCSs. Then, in Sec. 3.2 we describe the fixpoint listing framework and in Sec. 3.3 we instantiate it for our setting by defining a suitable closure operator. In Sec. 3.4 we also show that this operator is a generalisation of the closure operator which was defined within the same algorithmic framework for the case of MCCSs [14] and in Sec. 3.5 we discuss implementation details.

3.1 The Applicability of the Fixpoint Listing Framework

The divide-and-conquer fixpoint listing algorithm enumerates all closed sets of a closure operator from a given set system. A **set system** is a family of subsets $\mathcal{F} \subseteq \mathcal{P}(A)$ over some ground set A, where $\mathcal{P}(A)$ the power set of A. A set $F \in \mathcal{F}$ is called *closed* if it is a fixpoint of some **closure operator** $\rho : \mathcal{F} \to \mathcal{F}$, i.e., $\rho(F) = F$. For the operator ρ to be a closure operator it must satisfy three properties: extensivity ($F \subseteq \rho(F)$ for all $F \in \mathcal{F}$); monotonicity ($\rho(F) \subseteq \rho(F')$ for all $F, F' \in \mathcal{F}$ with $F \subseteq F'$); and idempotence ($\rho(\rho(F)) = \rho(F)$ for all $F \in \mathcal{F}$).

In order for the fixpoint algorithmic framework to be applicable for mining N-MCCSs, the set of closed N-CCSs needs to be a superset of the set of N-MCCSs. This is guaranteed by the properties of the closure operator. If an N-CCSs F is maximal it means that it cannot be extended by any other entity. From the extensivity of the closure operator and from the fact that $\rho(F) \in \mathcal{F}$, if follows that $\rho(F) = F$. Which means that F is a closed N-CCS.

In order for the framework to be applicable for mining the closed sets of a set system, this set system must satisfy a particular property called *strong accessibility* [2]. We now prove that independent of the input data $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$ the corresponding **set system of N-CCSs**, defined as

$$\mathcal{F}_{\mathbb{D}} = \{F \subseteq E \mid F \text{ connected } \wedge F \text{ complete}\}$$

is always strongly accessible. For a set system $\mathcal{F} \subseteq \mathcal{P}(A)$ and a set $F \in \mathcal{F}$, let us denote by $Aug(F) = \{a \in A \mid F \cup \{a\} \in \mathcal{F}\}$ the set of valid **augmentation elements** of F. Then $\mathcal{F}$ is called **strongly accessible** if for all $X \subset Y \subseteq A$ with $X, Y \in \mathcal{F}$ there is an element $e \in (Aug(X) \setminus X) \cap Y$.

Theorem 1. *For all relational data $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$, the set system $\mathcal{F}_{\mathbb{D}}$ of N-CCSs is strongly accessible.*

In [14] we showed that for multi-relational data containing just binary relationships the set system of Complete Connected Subsets (CCSs) is strongly accessible. The same argumentation line can be used to prove Theorem 1 as well.

3.2 N-RMiner

As we have already established that the divide and conquer fixpoint listing algorithmic framework of [2] is applicable to the case of N-CCSs, we now describe the instantiation of the framework for this setting.

Algorithm 1. N-RMiner: List all N-MCCSs

```
N-RMiner($F, B$)
 1: Select $e \in Aug(F) \setminus (F \cup B)$
 2: $F' = g(F \cup \{e\})$
 3: if $F' \cap B = \emptyset$ then
 4:     if $F' = Aug(F')$ then
 5:         Output $F'$
 6:     else
 7:         N-RMiner($F'$, $B$)
 8:     end if
 9: end if
10: N-RMiner($F$, $B \cup \{e\}$)
```

The general structure of this divide-and-conquer algorithm is shown in Algorithm 1. In each recursive call, the algorithm selects an element e from the set of valid augmentation elements, $Aug(F)$, of the current solution F(line 1), and splits the search space into two subtrees: one subtree in which all N-CCSs include the element e (lines 3-9) and another subtree in which all N-CCSs exclude e (line 10). This is achieved by adding it to B, which represents the set of elements already considered as extensions to F (line 10). The fact that only closed patterns are sought is ensured in line 2, where the expanded set $F \cup \{e\}$ is potentially further expanded by applying the operator g (see bellow for a definition).

As N-RMiner enumerates all closed N-CCSs, we added lines 4-5 to ensure it outputs N-MCCSs only, i.e., N-CCSs F for which there are no augmentation elements not yet in F. Formally this is verified by checking if $F = Aug(F)$.

As defined in Sec. 3.1, the set of augmentation elements $Aug(F)$ of a set $F \in \mathcal{F}_{\mathbb{D}}$ from a set system is the set of all elements that can be individually added to F to yield another set from the same set system. Specifically for the set system $\mathcal{F}_{\mathbb{D}}$ of N-CCSs, and given a relational database $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$, the set $Aug(F)$ corresponds to the following set: $Aug(F) = \{e \in E \mid F \cup \{e\} \text{ is complete and connected}\}$.

In [14] we presented a useful additional feature of the algorithm which gives the option of focusing on a smaller set of patterns by defining additional constraints on the pattern syntax. We defined constraints on the minimum number of entities per entity type (minimum coverage constraint) as well as an upper bound which can safely be used for pruning as the set system still remains strongly accessible. The same type of constraints can be used here exactly in the same way.

3.3 The Closure Operator

Since Algorithm 1 is based on adding elements from $Aug(F)$ one by one, it is important to introduce the notion of compatibility of an element with a set, upon which the definition of the closure operator g is based.

Definition 4. *For some relational data $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$, we say that an entity e is* ***compatible*** *with a set $G \subseteq E$ and denote $e \,\|\, G$, iff all critical sets $F' \subseteq (G \cup \{e\})$ with $e \in F'$, are covered in $\mathcal{R}$.*

We call the set of all $e \in E$ that are compatible with a set $F \in \mathcal{F}_{\mathbb{D}}$ the **set of compatible entities** of F and denote it as $Comp(F)$. We note here that the compatibility property is anti-monotone. This means that for $F, F' \subseteq E$ with $F \subseteq F'$, if for $e \in E$ it holds that $e \,\|\, F'$ then $e \,\|\, F$. Therefore also for $F, F' \in \mathcal{F}_{\mathbb{D}}$ with $F \subseteq F'$, $Comp(F') \subseteq Comp(F)$.

We would like to note here that for entities of type $i \notin \mathbf{t}(Aug(F))$ there are no critical sets between them and F and therefore by definition of compatibility $Comp(F) \cap E_i = E_i$.

Lemma 1. *Let $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$ some relational data. A set $F \subseteq E$ is complete if and only if for all $e \in F$ it holds that $e \,\|\, F$.*

Proof. If an $F \subseteq E$ is complete then all the critical subsets of F are covered in $\mathcal{R}$. Since $e \in F$, we have that all critical subsets of $F \cup \{e\}$ are covered, and therefore $e \,\|\, F$. Conversely, for an $F \subseteq E$ such that for all $e \in F$ it holds that $e \,\|\, F$, we have that for all $e \in F$ all critical subsets containing e are covered in $\mathcal{R}$, which means that all critical subsets of F are covered. Thus F is complete.

It follows from Lemma 1 that for a set $F \in \mathcal{F}_{\mathbb{D}}$, $F \subseteq Comp(F)$. It also follows that $Aug(F) \subseteq Comp(F)$, because for every element $e \in Aug(F)$, $e \,\|\, F$ always holds. Therefore $F \subseteq Aug(F) \subseteq Comp(F)$.

From Lemma 1 it also follows that for $F \in \mathcal{F}_{\mathbb{D}}$, $Comp(F)$ can equivalently be defined as the set of all $e \in E$ such that $F \cup \{e\}$ is complete.

We are now able to define the operator g. The definition of g requires that every element in the closure of F is compatible with all the compatible elements of F. Formally:

$$g(F) = F \cup \{e \in Aug(F) \setminus F : e \,\|\, Comp(F)\}, F \in \mathcal{F}_{\mathbb{D}}.$$

Before we show that g is a closure operator by proving each of the essential properties, we give an example of applying g to a set.

Example 4. Let us consider again the toy dataset of Fig. 1. Let us also consider that $F = \{$P2, Christos Faloutsos$\}$. Then $Aug(F) \setminus F = \{$P1, U1, U2, large graphs, algorithms, Jure Lescovec$\}$ and $Comp(F) = \{$P1, P2, U1, U2, large graphs, algorithms, Jure Lescovec, Christos Faloutsos$\}$ and $g(F) = \{$P2, Christos Faloutsos, Jure Lescovec$\}$. We see that from all the elements of $Aug(F) \setminus F$ only "Jure Lescovec" is compatible with $Comp(F)$ because all critical sets, namely $\{$P1,Jure Lescovec$\}$ and $\{$P2,Jure Lescovec$\}$, are covered.

Proposition 1. *For all $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$ the codomain of g is the set system $\mathcal{F}_{\mathbb{D}}$ of N-CCSs, i.e., for all $F \in \mathcal{F}_{\mathbb{D}}$ it holds that $g(F) \in \mathcal{F}_{\mathbb{D}}$.*

Proof. We need to show that for every $F \in \mathcal{F}_{\mathbb{D}}$, $g(F)$ is complete and connected. When $F = \emptyset$ then $g(F) = \emptyset$ and therefore it is complete and connected. We show this is the case as well for all $F \neq \emptyset$, $F \in \mathcal{F}_{\mathbb{D}}$.

We have that $g(F) \subseteq Aug(F)$. From the definition of the set $Aug(F)$ this means that for every $e \in g(F)$, $F \cup e$ is connected. This means that there is a sequence of pairwise related entities between every pair $e, f \in g(F)$. Therefore $g(F)$ is connected.

To show completeness let us assume that $g(F)$ is not complete. This means that there is a critical subset $F' \subseteq g(F)$, that is not covered. We have that F is complete since $F \in \mathcal{F}_{\mathbb{D}}$. This means that $F' \not\subseteq F$ and therefore there must be at least one $e' \in F'$ such that $e' \in \{e \in Aug(F) \setminus F : e \,\|\, Comp(F)\}$. We therefore have that all critical subsets of $Comp(F)$ that contain e' are covered. Since $F' \subseteq g(F) \subseteq Comp(F)$, the fact that F' is not covered is a contradiction. Therefore $g(F)$ is complete.

It is trivial to see that g is extensive, as it does not remove any elements from the set it is applied to. Next we will prove that g is also monotone and idempotent (as long as the data satisfies some properties).

Proposition 2. *For all $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$ the operator g is monotone.*

Proof. Assume the operator is not monotone, i.e., there are $F', F \in \mathcal{F}_{\mathbb{D}}$ with $F \subseteq F'$ such that $g(F) \not\subseteq g(F')$. Hence, there is an $e \in g(F) \setminus g(F')$. We also have $e \in g(F) \setminus F$ because g is extensive. From the definition of g it follows $e \,\|\, Comp(F)$. By anti-monotonicity of compatibility, $F \subseteq F'$ implies that $Comp(F) \supseteq Comp(F')$. Applying once again the monotonicity of compatibility, $e \,\|\, Comp(F)$ also implies $e \,\|\, Comp(F')$. Since $F' \subseteq Comp(F')$ we have that $e \,\|\, F'$. Also $e \in (Aug(F) \setminus F)$ and since F' is a connected superset of F, e is connected to F', and thus $e \in Aug(F')$. Hence $e \notin g(F')$ is a contradiction.

Before we give the proposition about idempotence, we define a class of multi-relational data called **acyclic multi-relational data** as all $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$ such that there does not exist a sequence of entity types $\{i_1 \ldots i_l\}$ such that for $j = 1 \ldots l-1$, $\{i_j, i_{j+1}\} \subseteq R \in \mathbf{R}$ and $i_1 = i_l$.

Proposition 3. *For all acyclic $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$ with the property that $\nexists e \in E$ such that $\exists R \in \boldsymbol{R}$ with $e \,\|\, (\cup_{i \in R} E_i)$, the operator g is idempotent.*

Proof. Assume that for an acyclic $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$, such that $\nexists e \in E$ such that $\exists R \in \mathbf{R}$ with $e \parallel (\cup_{i \in R} E_i)$, the operator g is not idempotent. This means that $\exists f \in g(g(F))$ such that $f \notin g(F)$. This can happen if:

(a) $f \nparallel Comp(F)$. Since $f \in g(g(F))$, $f \parallel Comp(g(F))$. For all $e \in g(F)$ we have that $e \parallel Comp(F)$. This means that e is compatible with all the compatible elements of F including F itself. Therefore adding e to F does not change this set which means that for every $e \in g(F)$, $Comp(F \cup \{e\}) = Comp(F)$. Therefore $Comp(g(F)) = Comp(F)$ and as a result $f \parallel Comp(F)$. A contradiction.

(b) $f \notin Aug(F)$. From the anti-monotonicity of compatibility and $f \parallel Comp(F)$, it holds that $f \parallel F$. Therefore for $f \notin Aug(F)$ it must be that f is not connected to F. Therefore $\mathbf{t}(f) \notin \mathbf{t}(Aug(F))$ which means that $\mathbf{t}(f)$ belongs to a relationship type R of which no entity types are in F. However, $\mathbf{t}(f) \in \mathbf{t}(Aug(g(F)))$. Therefore there must be an $e \in g(F)$ such that there is a set F', with $\mathbf{t}(F') \subseteq R \in \mathbf{R}$, $e, f \in F'$ and $F' \in \mathcal{R}$. Since $e \in g(F)$ we have that $e \parallel Comp(F)$. Since $\mathbb{D}$ is acyclic, no entity type of R except $\mathbf{t}(e)$ belongs to $Aug(F)$. Therefore we have that for every $i \in R$, $i \neq \mathbf{t}(e)$, $Comp(F) \cap E_i = E_i$. Therefore, for $e \parallel Comp(F)$ it must be that $e \parallel (\cup_{i \in R} E_i)$ (since e is compatible to the entities of the same type as well). A contradiction.

From the above propositions we have the following corollary:

Corollary 1. *For all acyclic multi-relational data* $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$ *with the property that* $\nexists e \in E$ *such that* $\exists R \in \boldsymbol{R}$ *with* $e \parallel (\cup_{i \in R} E_i)$*, the operator* g *is a closure operator.*

We note here that even though the idempotency of the operator g holds for acyclic multi-relational data that do not contain any entity belonging to all the relationship instances of a relationship, our method can still be applied to any multi-relational data by introducing an operator applying g to a set F as many times as required until the mapped set does not increase any more.

3.4 The Generality of the Closure Operator

The fixpoint listing algorithmic framework has also been used for mining MCCSs in multi-relational data with binary relationships. The instantiated algorithm for this setting is called RMiner [14]. Here we are going to show that the closure operator g is more general than the closure g_2 defined in R-Miner as:

$$g_2(F) = \{e \in Aug(F) : Comp'(F \cup \{e\}) = Comp'(F)\}$$

The set of compatible elements for a set $F \in \mathcal{F}_{\mathbb{D}}$ is defined in [14] as $Comp'(F) = \{e \in E \mid F \cup \{e\} \text{ is complete}\}$. In the conclusions of Lemma 1 we said that this is an equivalent definition for the set *Comp* that we defined in this paper. Therefore *Comp* and *Comp'* map to the same set.

Following from that the pattern syntax of N-MCCSs is a generalisation of the pattern syntax of MCCSs it also holds that the codomain of the two closure operators is the same. We can now state the following theorem.

Theorem 2. *For all relational data $\mathbb{D} = (E, \boldsymbol{t}, \mathcal{R}, \boldsymbol{R})$, such that $\forall R \in \boldsymbol{R}$, $|R| = 2$, the closure operator $g : \mathcal{F}_\mathbb{D} \to \mathcal{F}_\mathbb{D}$ is a generalisation of the closure operator $g_2 : \mathcal{F}_\mathbb{D} \to \mathcal{F}_\mathbb{D}$ [14], i.e., $\forall F \in \mathcal{F}_\mathbb{D}$, $g(F) = g_2(F)$.*

Proof. We are first going to prove that $g(F) \supseteq g_2(F)$. Let us assume the opposite, i.e., $\exists e \in g_2(F)$ such that $e \notin g(F)$. $e \notin g(F)$ means that $e \notin F$ and $e \nparallel Comp(F)$. Therefore there is at least one $e' \in Comp(F)$ such that $\{e, e'\} \notin \mathcal{R}$ which means that $Comp(F \cup \{e\}) \subset Comp(F)$. Contradiction.

Now we are going to prove that $g(F) \subseteq g_2(F)$. Let us assume the opposite, i.e., that $\exists e \in g(F)$ such that $e \notin g_2(F)$. $e \notin g_2(F)$ means that $Comp(F \cup \{e\}) \subset Comp(F)$ and therefore there exists at least one $e' \in Comp(F)$ such that $\{e, e'\} \notin \mathcal{R}$. This means that $e \nparallel Comp(F)$ which is a contradiction since we have assumed that $e \in g(F)$. Therefore $g(F) = g_2(F)$.

The result presented in Theorem 2 is very important as it means that when N-RMiner is applied to data with binary relationships only, it enumerates exactly the same set of patterns as RMiner which is using the operator g_2.

3.5 Implementation Details

Our implementation is based on giving an id to every relationship instance and storing the relationship instance ids, in a global structure containing for every entity and every relationship type, a list of relationship instance ids. Then the coverage of a critical subset can easily be checked by checking whether the intersection of the relationship instance ids of the entities it contains is non-empty. Although to check the compatibility of an element with a set all critical sets need to be checked for coverage, it can be done more efficiently as it suffices for one critical set not to be covered in order for the compatibility not to hold.

The choice of the next element to extend the current partial solution is implemented with a for loop using an ordering of the entities in terms of increasing $Aug(e)$. This way the number of closure checks is reduced.

4 Assessment of Patterns

We assess the quality of a pattern based on a subjective interestingness measure which formalises how interesting a pattern is based on the prior information of the user. We adopt a definition of interestingness which was proposed for binary tables and adapted for binary related multi-relational data as well in our previous work [4,13]. According to this definition interestingness is defined as unexpectedness with respect to a Maximum Entropy model of the prior information of the user. In what follows we describe the type of prior information we consider in this paper and formalize and derive the Maximum Entropy optimization problem. We then give the definition of the interestingness measure.

4.1 The Maximum Entropy Model

In this paper we consider as prior information the number of relationship instances an entity participates in. This intuitively means that a pattern containing entities that appear less frequently in relationship instances will be deemed more interesting by our measure. As an example consider a movie database consisting of three entity types, "person", "role" and "movie" and a 3-ary relationship between them. According to this prior information, a pattern containing persons that appear in many films under any role will be rendered less interesting than a pattern containing people that appear less often as the later is more unexpected.

We formalize this prior information by fitting the Maximum Entropy (MaxEnt) distribution P on the data, with constraints on the expected sum of the relationship instances every entity participates in, being equal to the actual sum of relationship instances it participates in. Without details, we point out that a similar reasoning as in [13,14] establishes that this MaxEnt model P is a product distribution, with a factor P_R for each relationship type—so here we focus on how to obtain a MaxEnt model for just one n-ary relationship type.

For convenience, let us represent each n-ary relationship $\mathcal{R}_R$ (with $|R| = n$) of some multi-relational data $\mathbb{D} = (E, \mathbf{t}, \mathcal{R}, \mathbf{R})$ by means of an n-dimensional binary valued tensor $\mathbf{D}_R$. The dimensions of $\mathbf{D}_R$ are indexed by the entity types $t \in R$, and each dimension itself is indexed by an index i_t running over all entities of type $t \in R$. We denote as e_{i_t} the i'th entity of type t. The relationship $\mathcal{R}_R$ is encoded in $\mathbf{D}_R$ by ensuring that $\mathbf{D}_R(i_1, \ldots, i_n) = 1$ iff $\{e_{i_1}, \ldots, e_{i_n}\} \in \mathcal{R}_R$. Then the prior information on the relationship type R can be expressed as:

$$\sum_{\mathbf{D}_R} P_R(\mathbf{D}_R) \sum_{(i_1 \ldots i_n): i_t = k} \mathbf{D}_R(i_1, \ldots, i_t, \ldots, i_n) = d_R^{t,k}, (\forall t \in R)$$

where $d_R^{t,k}$ is the sum of all elements in $\mathbf{D}_R$ for which $i_t = k$ or in other words the number of relationship instances that the kth entity participates in. Finally, being a probability distribution, P_R needs to be normalised: $\sum_{\mathbf{D}_R} P_R(\mathbf{D}_R) = 1$.

The MaxEnt optimization problem maximizes $\sum_{\mathbf{D}_R} -P_R(\mathbf{D}_R) \log(P_R(\mathbf{D}_R))$, the entropy, with respect to P_R, subject to the above constraints. Computing the Karush Kuhn Tucker optimality conditions, similar to [4], shows that the optimal solution is of the form:

$$P_R(\mathbf{D}_R) = \prod_{i_1 \ldots, i_n} P_R^{\{i_1, \ldots, i_n\}}(\mathbf{D}_R(i_1, \ldots, i_n)),$$

$$\text{where } P_R^{\{i_1, \ldots, i_n\}}(d) = \frac{\exp\left(d \cdot \sum_t \lambda_R^{t,i_t}\right)}{1 + \exp\left(\sum_t \lambda_R^{t,i_t}\right)}.$$

Here, λ_R^{t,i_t} is a Lagrange multiplier corresponding to the constraint for entity i_t of type t. Thus, every factor P_R of the distribution P, is itself a product of independent Bernoulli distributions for the elements of the tensor. The optimal value of the Lagrange multipliers can be found by solving the convex Lagrange dual optimization problem ([4] gives details for the binary case).

4.2 The Interestingness of a Pattern

The proposed interestingness measure is a trade-off between the self-information of a pattern, which measures how much information a pattern carries, and the description length of a pattern, which measures how concisely it conveys this information. Intuitively an end user wants to see patterns that convey as much information as possible, as concisely as possible. Thus, interestingness of an N-MCCS $F \in \mathcal{F}_{\mathbb{D}}$ is defined as follows:

$$\text{Interestingness}(F) = \frac{\text{SelfInformation}(F)}{\text{DescriptionLength}(F)}.$$

Let us define the set of maximal critical subsets of F, denoted as $\mathcal{M}(F)$, as the set of all $F' \subseteq F$ such that $|\mathbf{t}(F')| = |F'|$ and $\mathbf{t}(F') \subseteq R \in \mathbf{R}$ and $\nexists F'' \subseteq F$ with $\mathbf{t}(F') \subset \mathbf{t}(F'') \subseteq R$. The self-information of a pattern F is given by:

$$\text{SelfInformation}(F) = -\sum_{R \in \mathbf{R}} \sum_{F' \in \mathcal{M}(F)} \log(P_R(F')).$$

where $P_R(F')$ signifies the probability that the entities in F' are related in R.

For the terms with $\mathbf{t}(F') = R$, the probability $P_R(F')$ can be computed directly using the MaxEnt model as $P_R(F') = P_R^{F'}(1)$, i.e. the probability that the relationship instance specified by the entities in F' is present in $\mathcal{R}_R$.

For the terms with $\mathbf{t}(F') \subset R$, the probability that there *exists* at least one relationship instance of type R involving all entities from F' is computed as $P_R(F') = 1 - \prod_{I=\{i_t\}_{t \in R \setminus \mathbf{t}(F')}} \left(1 - P_R^{F' \cup I}(1)\right)$. This follows directly from the independence of the different entries in the tensor $\mathbf{D}_R$, under the MaxEnt model.

We define the description length of a pattern $F \in \mathcal{F}_{\mathbb{D}}$ by specifying for each entity whether it does or it does not belong to the pattern. We use $-log(p)$ bits to specify that an entity belongs to the pattern and $-log(1-p)$ bits to specify that it does not, where p is a probability parameter. The description length of a pattern is given by the following equation:

$$\text{DescriptionLength}(F) = -\sum_{i \notin F} \log(1-p) - \sum_{i \in F} \log(p).$$

5 Experiments

In this section we present a qualitative evaluation of our method, by discussing some of the top ranked patterns on real-world data sets and a quantitative evaluation by showing scalability results and comparisons with other methods. For all the experiments we used parameter p equal to the density of the dataset.

5.1 Qualitative Evaluation

Here we show the highest ranked patterns found in real world datasets. We performed experiments on two real world datasets, one created from Bibsonomy [1] (*bibsonomy-journals*) and another one created from IMDB[1](*imdb-3years*). Figure 2 depicts the diagrams of these datasets showing how different entity types are related as well as the number of instances for every relationship type and every entity type. To create the *bibsonomy-journals* dataset we selected the bibtex entries that corresponded to journal papers, along with their authors, users and tags. To create the *imdb-3years* dataset we created a view of the IMDB database containing person names and roles, as well as genres of movies produced from year 2008 to year 2010 and are not of the genre "Short". The role of a person in a movie can be actor, actress, producer, director etc.

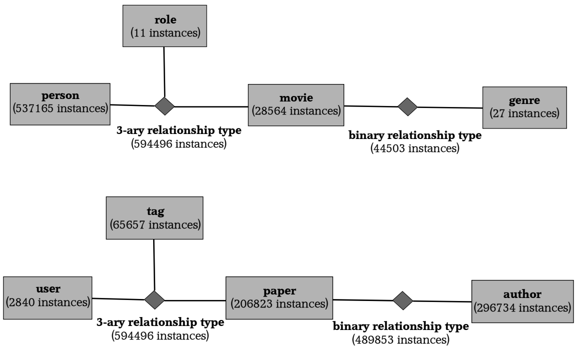


Fig. 2. Diagrams of the *imdb-3years* (top) and *bibsonomy-journals* datasets

We run experiments on both these datasets using different constraints. The results were ranked based on the interestingness measure we introduced in Sec. 4.2. In order to avoid redundancy we iteratively output the top ranked pattern and then re-rank the pattern set by taking into account in the interestingness of a pattern only relationship instances that have not been presented before.

The first ranked pattern on the *imdb-3years* dataset when requiring at least 2 papers, 2 titles, 2 roles and 1 genre, contains as entities: **2 persons:** Joel Coen; Ethan Coen; **4 titles:** The Yiddish Policemen's Union; True Grit; A Serious Man; Burn After Reading; **2 roles:** Director; Producer; **1 genre:** Drama. This pattern ranks first as it contains a lot of information (20 of relationship instances) conveyed in a concise way (9 entities). Recall here that our measure of self-information is an additive measure over the number of relationship instances and our measure of description length is an additive measure over the entities. The second pattern contains 18 relationship instances conveyed with 9 entities.

In the first experiment on *bibsonomy-journals* we required at least 2 entities from each entity type. The top pattern contains: **3 users:** 650753; 594907; 576800; **5 tags:** meningococcal; meningitidis; infections; neisseria; human; **6 papers:** (not shown here due to space constraints); **2 authors:** Heike Claus; Matthias Frosch. This pattern ranks highly as it conveys a lot of information

[1] See `http://www.imdb.com/`

(102 relationship instances) in a concise way (16 entities). The second pattern, in comparison, contains 48 relationship instances and 12 entities. We also run an experiment not necessarily requiring any entities of the entity type *user*, but at the same time requiring at least 5 tags, 5 papers and 2 authors. The most interesting pattern remains the same as in the previous experiment.

5.2 Quantitative Evaluation

In this subsection we show the performance of N-RMiner by presenting scalability results as well as results comparing N-RMiner with other algorithms on special cases of multi-relational data. All the experiments were run on a CentOS Linux machine with Intel Xeon@2.6GHz and 24GB of RAM.

In order to test the scalability we used the *bibsonomy-journals* dataset, of which we got five snapshots of increasing size. The first four snapshots were produced by randomly picking 0.01%, 0.1%, 1% and 10% of the entities of the type "paper" and selecting the entities of the other types that correspond to them. The final snapshot contains all of the *bibsonomy-journals* dataset. The results are shown in Table 1. The "-"s mean that the particular run did not finish within 2 days. As with most local pattern mining algorithms, time scales exponentially to the input size. However, when using constraints of at least 2 entities per entity type N-RMiner runs within a few hours for data sizes of approximately 5×10^5. The space that N-RMiner consumes also increases exponentially to the input size. However it only grows up to 1.5 GB for the largest dataset. As we are unaware of any other local pattern mining method that considers the general case of multi-relational data, where there can be any number of relationships of any arity, we compare N-RMiner with methods for specific cases of multi-relational

Table 1. Scalability testing of N-RMiner for increasing subsets of *bibsonomy-journals*

Constraints	#Entities	#N-MCCSs	Time (sec)	Space (MB)	Depth	Size of max N-MCCS
(0,0,0,0)	63	17	0	1.42	3	13
	1,167	295	2.17	3.63	5	100
	9,643	4,177	2599.22	28.32	14	800
	77,452	-	-	-	-	-
	567,416	-	-	-	-	-
(1,1,1,1)	63	12	0	1.40	1	13
	1,167	206	0.11	3.24	4	100
	9,643	2,186	8.52	18.99	5	83
	77,452	22,267	1,055.14	175.68	11	107
	567,416	-	-	-	-	-
(2,2,2,2)	63	0	0	1.37	0	0
	1,167	0	0.01	3.01	0	0
	9,643	0	0.6	16.98	0	0
	77,452	3	118.36	151.25	5	14
	567,416	359	13,121.4	1,502.95	16	27

data, namely Data-Peeler that mines n-Sets in one n-ary relationship ($n > 2$) [3] and RMiner which mines MCCSs in multi-relational data with binary relationships [14]. As we have shown in Sec. 2.3, both these pattern syntaxes constitute special cases of N-MCCSs and therefore the comparison of the algorithms is fair.

Table 2 (left) shows the time comparison between N-RMiner and Data-Peeler on random snapshots of increasing sizes of the 3-ary relationship of the *bibsonomy-journals* dataset. Data-Peeler offers the opportunity to use constraints on the minimum number of entities per entity type, like ours. However n-Sets are not as flexible as N-MCCSs in the sense that all entity types of the relationship need to be present in a pattern. We therefore compare the two algorithms in the simplest common setting of requiring at least one entity per entity type. The results show that N-RMiner outperforms Data-Peeler by at least one order of magnitude most of the times. Also for the dataset of 36,367 entities Data-Peeler could not run as it was crashing due to high memory requirements. This is because Data-Peeler uses a dense representation to store the dataset.

Table 2. Comparing N-RMiner w. Data-Peeler (left) and N-RMiner w. RMiner (right).

Cons-traints	#Entities	N-RMiner time (sec)	Data-Peeler time (sec)
(1,1,1)	39	0	0
	774	0.28	2.39
	5,405	97.75	2960.73
	36,367	182,156	-

Cons-traints	#Entities	N-RMiner time (sec)	RMiner time (sec)
(1,1,1)	3,291	2.81	1.65
	8,686	21.41	12.01
	51,203	1,296	610
	111,320	7,456	2,813

Table 2 (right) compares running times between N-RMiner and RMiner on increasing subsets from IMDB containing two binary relationships: between directors and titles, and between titles and genres. Since both algorithms use the same algorithmic framework [2] and the closure operator of N-RMiner is a generalisation of RMiner's (see Sec 3.2), it should be no surprise that they are similarly efficient, with a small slowdown of N-RMiner owing to its greater generality.

6 Conclusion

In this paper we took an important step towards the development of local pattern mining algorithms that can be directly applied to data as it often presents itself in real-life: represented in a relational database. More specifically, the N-MCCS pattern syntax and associated mining algorithm N-RMiner are applicable to multi-relational data with categorical attributes and n-ary relationships. Our approach is a strict generalisation of two less general problem settings [13,14,3], and it matches or even outperforms algorithms that were designed for these special cases. Additionally, we introduced a subjective interestingness measure for ranking the N-MCCSs found. An interesting research direction further increasing generality would be the extension of our results to ordinal and numerical data.

Acknowledgements. We thank L. Cerf for his help in running Data-Peeler. This work is funded by the grant EP/G056447/1 and by the PASCAL2 NoE.

References

1. Knowledge and data engineering group, University of Kassel: Benchmark folksonomy data from bibsonomy, version of January 1 (2012)
2. Boley, M., Horvath, T., Poigné, A., Wrobel, S.: Listing closed sets of strongly accessible set systems with applications to data mining. Theoretical Computer Science 411(3), 691–700 (2010)
3. Cerf, L., Besson, J., Robardet, C., Boulicaut, J.-F.: Closed patterns meet n-ary relations. ACM Trans. Knowl. Discov. Data 3(1), 3:1–3:36 (2009)
4. De Bie, T.: Maximum entropy models and subjective interestingness: an application to tiles in binary databases. Data Mining and Knowledge Discovery 23(3), 407–446 (2011)
5. Dehaspe, L., Toivonen, H.: Discovery of frequent datalog patterns. Data Mining and Knowledge Discovery 3, 7–36 (1999)
6. Elmasri, R., Navathe, S.B.: Fundamentals of Database Systems. Addison Wesley (2006)
7. Goethals, B., Page, W.L., Mampaey, M.: Mining interesting sets and rules in relational databases. In: Proc. of the ACM Symposium on Applied Computing (SAC), pp. 997–1001 (2010)
8. Jaschke, R., Hotho, A., Schmitz, C., Ganter, B., Stumme, G.: Trias–an algorithm for mining iceberg tri-lattices. In: Proc. of the Sixth International Conference on Data Mining (ICDM), pp. 907–911 (2006)
9. Ji, L., Tan, K.-L., Tung, A.K.H.: Mining frequent closed cubes in 3d datasets. In: Proc. of the 32nd International Conference on Very Large Data Bases (VLDB), pp. 811–822 (2006)
10. Koopman, A., Siebes, A.: Discovering relational item sets efficiently. In: Proc. of the SIAM Conference on Data Mining (SDM), pp. 108–119 (2008)
11. Ng, E.K.K., Ng, K., Fu, A.W.-C., Wang, K.: Mining association rules from stars. In: Proc. of the 2002 IEEE Int. Conference on Data Mining, ICDM, pp. 322–329 (2002)
12. Nijssen, S., Kok, J.N.: Efficient frequent query discovery in FARMER. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS(LNAI), vol. 2838, pp. 350–362. Springer, Heidelberg (2003)
13. Spyropoulou, E., De Bie, T.: Interesting multi-relational patterns. In: Proc. of the IEEE International Conference on Data Mining, ICDM, pp. 675–684 (2011)
14. Spyropoulou, E., De Bie, T., Boley, M.: Mining interesting patterns in multirelational data. Data Mining and Knowledge Discovery (in print, 2013)

Learning Hierarchical Multi-label Classification Trees from Network Data

Daniela Stojanova[1], Michelangelo Ceci[2], Donato Malerba[2], and Sašo Džeroski[1,3,4]

[1] Jožef Stefan Institute, Department of Knowledge Technologies, Ljubljana, Slovenia
[2] Dipartimento di Informatica, Università degli Studi di Bari, Bari, Italy
[3] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
[4] COE for Integrated Approaches in Chemistry and Biology of Proteins, Slovenia
{daniela.stojanova,saso.dzeroski}@ijs.si, {ceci,malerba}@di.uniba.it

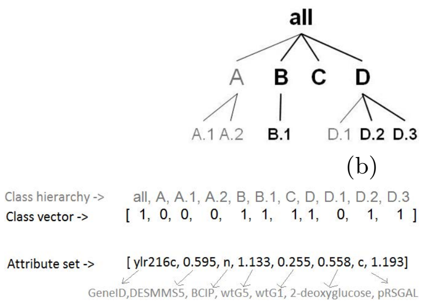
Abstract. We present an algorithm for hierarchical multi-label classification (HMC) in a network context. It is able to classify instances that may belong to multiple classes at the same time and consider the hierarchical organization of the classes. It assumes that the instances are placed in a network and uses information on the network connections during the learning of the predictive model. Many real world prediction problems have classes that are organized hierarchically and instances that can have pairwise connections. One example is web document classification, where topics (classes) are typically organized into a hierarchy and documents are connected by hyperlinks. Another example, which is considered in this paper, is gene/protein function prediction, where genes/proteins are connected and form protein-to-protein interaction (PPI) networks. Network datasets are characterized by a form of autocorrelation, where the value of a variable at a given node depends on the values of variables at the nodes it is connected with. Combining the hierarchical multi-label classification task with network prediction is thus not trivial and requires the introduction of the new concept of network autocorrelation for HMC. The proposed algorithm is able to profitably exploit network autocorrelation when learning a tree-based prediction model for HMC. The learned model is in the form of a Predictive Clustering Tree (PCT) and predicts multiple (hierarchically organized) labels at the leaves. Experiments show the effectiveness of the proposed approach for different problems of gene function prediction, considering different PPI networks. The results show that different networks introduce different benefits in different problems of gene function prediction.

1 Introduction

Hierarchical multi-label classification (HMC) is a variant of the classification task where instances may belong to multiple classes at the same time and classes are organized in a hierarchy. Due to the large number of applications (e.g., computational biology [31] and text categorization [7]) this particular classification setting has recently attracted the attention of many researchers.

Another variant of the classification task which, due to the large number of applications, has attracted the attention of researchers over the last few years is that of learning predictive models from network data. This is due to the ubiquity of networks, which can be found in in several social, economical and scientific fields. As recently recognized [29], this problem is not trivial, mainly because the network introduces some form of autocorrelation which is a direct violation of the assumption that data are independently and identically distributed (i.i.d.). However, autocorrelation also offers a unique opportunity to improve the performance of predictive models on network data, as inferences about one entity can be used to improve inferences about related entities.

A. METABOLISM
A.1 amino acid metabolism
A.2 nitrogen, sulfur, selenium met.
A.1.3 assimilation of ammonia
A.1.3.1 metabolism of glutamine
A.1.3.1.1 biosynthesis of glutamine
A.1.3.1.2 degradation of glutamine
...
B. ENERGY
B.1 glycolysis and gluconeogenesis
C. CELL CYCLE and DNA PROCESSING
D. TRANSCRIPTION
D.1 RNA synthesis
D.2 RNA processing
D.3 transcriptional control


all
A
B
C
D
A.1 A.2
B.1
D.1 D.2 D.3
(b)
Class hierarchy -> all, A, A.1, A.2, B, B.1, C, D, D.1, D.2, D.3
Class vector -> [1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1]
Attribute set -> [ylr216c, 0.595, n, 1.133, 0.255, 0.558, c, 1.193]
GeneID, DESMMS5, BCIP, wtG5, wtG1, 2-deoxyglucose, pRSGAL

(a) (c)

Fig. 1. An illustration of the task of HMC. (a) A part of the FUN hierarchy [21]. (b) An example of hierarchial labeling of instances (subset) of the hierarchy. (c) The class vector (top) and the attribute vector (bottom) of the instance.

The combination of these two research directions paves the way to solving prediction problems where instances may belong to multiple classes at the same time, classes are organized in a hierarchy and instances may be connected according to a network structure. One of the main cases where this combination turns out to be beneficial is in gene function prediction, where the goal is to discover the biological functions of the genes/proteins. In fact, in this case, ontologies and catalogs of functions of genes/proteins such as the Gene Ontology (GO) and MIPS-FUN assume that functional classes are organized hierarchically and that general functions include more specific functions.

Besides these relationships among classes, it is also possible to identify relationships among examples. An example is that of protein-protein interaction (PPI) networks which represent correlations between genes/proteins. Indeed, the topic of using protein-protein interaction (PPI) networks in the identification and prediction of protein functions has attracted increasing attention in recent years. The motivation for this stream of research is best summarized by the statement that "when two proteins are found to interact in a high throughput assay, we also tend to use this as evidence of functional linkage" [17].

This paper demonstrates the benefits (in terms of predictive accuracy) of considering network information in multi-label gene function prediction. In particular, during the learning process, we identify and take into account network autocorrelation, that is, the statistical relationships between the same variable (e.g., protein function) on different but related (dependent) objects (e.g., interacting proteins) [29]. We present a tree-based algorithm which extends and revises the system CLUS-HMC [31] that learns Predictive Clustering Trees (PCTs) by considering network autocorrelation in the setting of Hierarchical Multi-label Classification.

The paper is organized as follows. In the next section, we present some related work. In Sections 3 and 4, we formally introduce the learning setting we intend to solve and present the proposed approach, namely NHMC (Network Hierarchical Multi-label Classification). In Section 5, we empirically evaluate the proposed algorithm on 12 yeast datasets using each of the MIPS-FUN and GO annotation schemes and exploiting 3 different PPI networks. Finally, we draw some conclusions and outline some directions for future work.

2 Related Work

Many machine learning approaches tackle the problem of multi-label classification on hierarchically-structured categories. A simple solution is to allow a classifier for a particular node to predict positive only if the classifier of its parent also predicts positive [3] [8]. A different approach is to construct a training set for each category such that it only consists of samples belonging to its parent [10]. Alternatively, large margin methods for structured output prediction can also be used [25]. In the work by Vens et al. [31] the algorithm CLUS-HMC is proposed. CLUS-HMC learns Predictive Clustering Trees (PCTs) for hierarchical multilabel classification. A similar approach is presented in Astikainen et al. [2], where a structured output kernel-based approach for enzyme function prediction is proposed. In [5], the authors propose to formulate the search for the optimal consistent multi-label as the finding (according to a greedy search) of the best subgraph in a tree/DAG.

As concerns the problem of learning predictive models from network data, numerous approaches have been designed for modeling a partially labeled network and providing accurate estimates of unknown labels associated with the unlabeled nodes. These approaches have been mainly studied in the research fields of collective inference. In collective inference, interrelated values are inferred simultaneously and estimates of neighboring labels influence one another [19,14,27]. In general, one of the major advantages of collective inference lies in its powerful ability to learn various kinds of dependency structures (e.g., different degrees of correlation [16]). However, as pointed out in [22], when the labeled data are very sparse, the performance of collective classification might be largely degraded due to the insufficient number of neighbors. This is overcome by incorporating informative "ghost edges" into the network to deal with sparsity issues [20,22]. An alternative solution to the sparsity of labels is provided by [6], where the authors resort to an active learning approach in order to judiciously select nodes for which a manual labeling is required from the domain expert. Finally, in [29] the authors exploit the concept of network autocorrelation as a heuristic to be used when learning PCTs.

Unfortunately, there are only few initial works that combine these two research directions and most of them are developed in the context of protein function annotation and prediction. In this context, some studies use PPI networks as one of the data sources. For example, Valentini [30] developed the *true-path rule* ensemble learner for genome-wide gene function prediction. In these ensembles, positive (negative) probabilistic predictions for a node transitively influence the ancestors (descendants) of the node. An empirical evaluation of both this method and Hierarchical Bayes [24], which operates in the same way by approximating the Bayesian-optimal predictor with respect to the H-loss, proved that the usage of hierarchical prediction methods results in a consistently improved performance as compared to methods which do not consider the structure of the reference ontology on protein functions. Information from PPI networks considered in this work is limited to binary (input) attributes which express the generic interaction of a gene with others. Outside the context of protein function prediction, in [18], the authors propose a multi-label collective classifier, which, however, does not consider the hierarchical organization of class labels.

3 Autocorrelation in Network HMC Tasks

In this Section, we first define the task of hierarchical multi-label classification (HMC). We next discuss the network setting that we consider in this paper.

3.1 The Task of HMC

For the HMC task, the input is a dataset consisting of example pairs $(x_i, y_i) \in \mathbf{X} \times 2^C$, where $\mathbf{X} = X_1 \times X_2 \ldots \times X_m$ is the space spanned by m attributes or features, while 2^C is the power set of $C = \{c_1, \ldots, c_K\}$, the set of all possible class labels. C is hierarchically organized with respect to a partial order $\preceq$ which represents the superclass relationship. Note that each y_i satisfies the *hierarchical constraint*:

$$c \in y_i \Rightarrow \forall c' \preceq c : c' \in y_i. \tag{1}$$

3.2 Network HMC

Following Steinhaeuser et al. [28], we view a training set as a single network of labeled nodes. Formally, the network is defined as an undirected edge-weighted graph *G=(V,E)*, where V is the set of labeled *nodes*, while $E \subseteq \{\langle u, v, w\rangle | u, v \in V, w \in \mathbb{R}^+\}$ is the set of *edges*, such that to each edge $u \leftrightarrow v$ is assigned a non-negative real number w, called the *weight* of the edge. It can be represented by a symmetric adjacency matrix $\mathbf{W}$, whose entries are positive ($w_{ij} > 0$) if there is an edge connecting i to j in G, and zero ($w_{ij} = 0$) otherwise. Edge weights can, for example express the strength of the interactions between proteins. Although the proposed method works with any non-negative weight values, we anticipate that in our experiments only binary (0/1) weights could be used, due to limitations of available data.

Each node of the network is associated with an example pair $(x_i, y_i) \in \mathbf{X} \times 2^C$, where $y_i = (y_{i_1}, y_{i_2}, ..., y_{i_q}), q \leq K$, is subject to the hierarchical constraint. Given a network $G = (V, E)$ and a function $\eta : V \longmapsto (\mathbf{X} \times 2^C)$ which associates each node with the corresponding example pair, we interpret the task of HMC as building a PCT which represents a multi-dimensional predictive function $f : \mathbf{X} \longmapsto 2^C$ whose prediction satisfies the hierarchical constraint (Formula (1)). When learning f, we take into account (e.g., maximize) the autocorrelation of the observed (hierarchically organized) classes y_i for the network G, and minimizes the predictive error f on the training data $\eta(V)$. It is noteworthy that, although f is learned by taking the network structure into account, it does not take as input the network structure. This is due to our network setting, which is very much different from existing approaches to network classification and regression, where typically the descriptive information is in a tight connection to the network structure. The connections (edges in the network) between the data in the training/testing set are predefined for a particular instance, and are used to generate the descriptive information associated to the nodes of the network (as in [28]). Therefore, in order to predict the value of the response variable(s), besides the descriptive information one needs the connections (edges in the network) to the related/similar entities.

3.3 Autocorrelation in NHMC Tasks

Network autocorrelation for HMC is a special case of network autocorrelation [13]. It can be defined as the statistical relationship between observations of a variable (e.g., protein function) on distinct but related (connected) nodes in a network (e.g., interacting proteins). Autocorrelation is typically defined for real-world variables. In HMC, domain values of the variable form a hierarchy, such as the GO hierarchy for protein functions. Therefore, it is possible to define network autocorrelation at various levels of the hierarchy. The network (e.g., PPI network) can provide useful (and diversified) information for different classes (e.g., different protein functions) at different levels of the hierarchy.

To better explain this concept, we refer to Fig. 2 which shows the DIP Yeast network. Fig. 2 (a) represents the network so that the examples that are not connected are randomly arranged along the ellipse border. To show the density of the PPI interactions, Fig. 2 (b) represents the same network so that all examples are arranged along the ellipse border. Fig. 2 (c) and Fig. 2 (d) provide us a different view of Fig. 2 (b), where examples are grouped according to the first (Fig. 2 (c)) and second level (Fig. 2 (d)) of the FUN hierarchy.

Keeping in mind that all these graphs represent the same number of edges, from the comparison of Fig. 2 (b) and Fig. 2 (c) we can see that the edges "move" from the center of the ellipse towards the border. This is clearly due to autocorrelation, since the number of interactions between genes of the same class

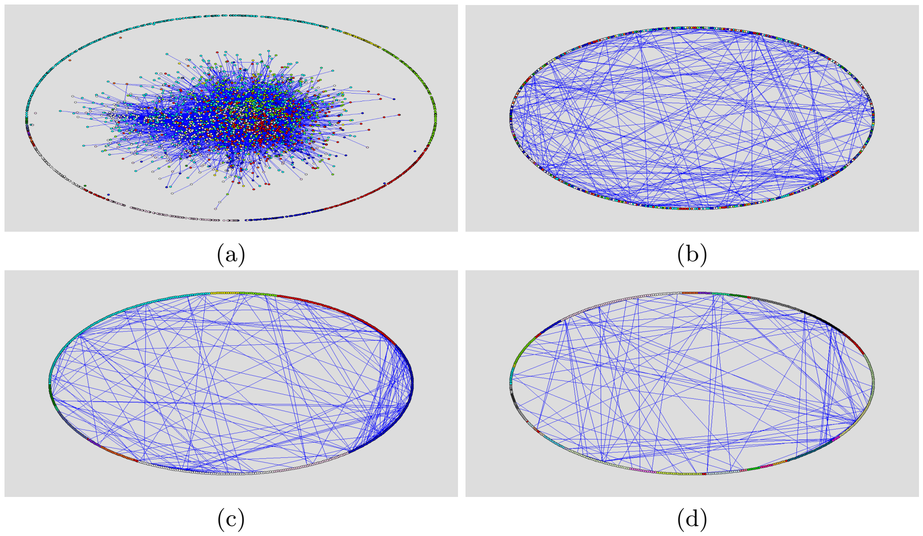


Fig. 2. DIP Yeast network. Different colors correspond to different classes of the FUN hierarchy. (a) Examples that are not connected are arranged along the ellipse's border; (b) Examples are arranged along the ellipse's border to show the density of the PPI interactions; (c) Examples are arranged along the ellipse's border and grouped according to the first level of the FUN hierarchy (not considering the root); d) Examples are arranged along the ellipse's border and grouped according to the second level of FUN. The networks are drawn by using the Pajek Software by Batagelj and Mrvar [4].

at the same level of the hierarchy is much larger than the number of interactions between genes of different classes. Moreover, by comparing Fig. 2 (c) and Fig. 2 (d) we notice that the autocorrelation effect is more localized at the second level of the hierarchy then at the first. Indeed, in Fig. 2 (d), we observe a reduction of the density of edges in the center of the ellipse (most of the edges overlap with (are hidden by) the examples arranged along the ellipse border).

4 NHMC

In this Section, we introduce the method NHMC (Network CLUS-HMC), which builds autocorrelation-aware HMC models. We shall start with a brief description of the algorithm CLUS-HMC which is at the base of NHMC. Moreover, before describing the NHMC method, we propose a new network autocorrelation measure for HMC tasks.

4.1 CLUS-HMC

The CLUS-HMC [31] algorithm builds HMC trees. These are very similar to classification trees, but each leaf predicts a hierarchy of class labels rather than a single class label. CLUS-HMC builds the trees in a top-down fashion as in classical tree induction algorithms, with a key difference in the search heuristics.

To select the best test in an internal node of the tree, the algorithm scores the tests according to the reduction in variance (defined below) induced on the set U of examples associated to the node. In CLUS-HMC, the variance is defined as follows: $Var(U) = \frac{1}{|U|} \cdot \sum_{u_i \in U} d(L_i, \overline{L})^2$, where $d(\cdot,\cdot)$ is a distance function on vectors associated to class labels of examples in U. Class labels associated to an example (x_i, y_i) in U are represented as a binary vector L_i of size $|C|$, such that $L_{i,k} = 1$ if $c_k \in y_i$, $L_{i,k} = 0$ otherwise. Obviously, each L_i has to satisfy the hierarchical constraint (Formula (1)). Finally, $\overline{L}$ is the average vector of $\{L_i\}_i$.

In the HMC context, class labels at higher levels of the annotation hierarchy are more important than class labels at lower levels. This is reflected in the distance measure used in the above formula, which is a weighted Euclidean distance: $d(L_1, L_2) = \sqrt{\sum_{k=1}^{K} \omega(c_k) \cdot (L_{1,k} - L_{2,k})^2}$, where $L_{i,k}$ is the k-th component of the class vector L_i and the class weights $\omega(c_k)$ decrease with the depth of the class in the hierarchy. More precisely, $\omega(c) = \omega_0 \cdot \text{avg}_j \{\omega(p_j(c))\}$, where $p_j(c)$ denotes the j-th parent of class c and $0 < \omega_0 < 1$). This definition of the weights allows us to take the label hierarchy into account: The hierarchy can be either a tree or a DAG, where we can have multiple parents of a single label.

For instance, consider the small hierarchy in Fig. 1(b), and two examples (x_1, y_1) and (x_2, y_2), where[1]:

$$y_1 = \{all, B, B.1, C, D, D.2, D.3\} \text{ and } y_2 = \{all, A, D, D.2, D.3\}$$

The vectors for y_1 and y_2 are: $L_1 = [1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1]$ and $L_2 = [1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1]$. The distance between the two is:

$$d([1,0,0,0,1,1,1,1,0,1,1],[1,1,0,0,0,0,0,1,0,1,1]) = \sqrt{3 \cdot w_0^2 + 4 \cdot w_0^3} \quad (2)$$

At each node of the tree, the test that maximizes the variance reduction is selected. This hopefully results in maximizing cluster homogeneity with respect to the target variable, as well as in improving the predictive performance of the tree. If no test can be found that significantly reduces variance (as measured by a statistical F-test), then the algorithm creates a leaf and labels it with a prediction, which can consist of multiple hierarchically organized labels.

We can now proceed to describe the top-down induction algorithm for building Network HMC trees. The search space is exactly the same as for CLUS-HMC, while the heuristics is different. The network is considered as background knowledge to exploit only in the learning phase.

4.2 Outline of the NHMC Algorithm

The top-down induction algorithm for building PCTs from network data is given below (Algorithm 1). It takes as input the network $G = (V, E)$ and the

[1] "all" indicates the root of the hierarchy.

corresponding HMC dataset U defined by applying $\eta\colon V \mapsto \mathbf{X} \times 2^C$ to the vertices of the network. It then recursively partitions U until a stopping criterion is satisfied (Algorithm 1 line 2). Since the implementation of this algorithm is based on the implementation of the CLUS-HMC algorithm, we call this algorithm NHMC (Network CLUS-HMC).

Algorithm 1. Top-down induction of NHMC

```
1: procedure NHMC($G, U$) returns tree
2: if stop($U$) then
3:     return leaf(Prototype($U$))
4: else
5:     $(t^*, h^*, \mathcal{P}^*) = (null, 0, \emptyset)$
6:     for each possible Boolean test $t$ according to the values of $X$ in $U$ do
7:         $\mathcal{P} = \{U_1, U_2\}$ partition induced by $t$ on $U$
8:         $h = \alpha \cdot \left(\frac{|U_1| \cdot A_Y(U_1) + |U_2| \cdot A_Y(U_2)}{|U|}\right) + (1 - \alpha) \cdot \left(Var'(U) - \frac{|U_1| \cdot Var'(U_1) + |U_2| \cdot Var'(U_2)}{|U|}\right)$
9:         if $(h > h^*)$ then
10:            $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
11:        end if
12:    end for
13:    $tree_1$ = NHMC($G, U_1$)
14:    $tree_2$ = NHMC($G, U_2$)
15:    return node($t^*, tree_1, tree_2$)
16: end if
```

Geary's C for HMC. In order to measure the autocorrelation of the response variable Y in the network setting for HMC, we propose a new statistics, named $A_Y(U)$, whose definition draws inspiration from Global Geary's C [15], originally defined for spatial data analysis.

Let $(x_i, y_i) \in U \subseteq \mathbf{X} \times 2^C$ be an example pair in a training set U of N examples. Let K be the number of classes in C, possibly defining a hierarchy. Let $d(L_i, L_j)$ be a distance measure defined for two binary vectors associated to two example pairs (x_i, y_i), (x_j, y_j). It can be any distance which can take the class-label hierarchy into account. We will use the distance defined above.

The autocorrelation measure $A_Y(U)$ is defined as follows:

$$A_Y(U) = 1 - \frac{(N-1) \cdot \sum_i \sum_j w_{ij} \cdot d(L_i, L_j)^2}{4 \cdot \sum_i \sum_j w_{ij} \quad \cdot \quad \sum_p d(L_p, \overline{L})^2} \tag{3}$$

The constant 4 in the denominator is included for scaling purposes. The new autocorrelation measure $A_Y(U)$ takes values in the unit interval $[0, 1]$, where 1 (0) means strong positive (negative) autocorrelation and 0.5 means no autocorrelation.

Heuristics. The major difference between NHMC and CLUS-HMC is in the heuristics we use for the evaluation of each possible split. The variance reduction heuristics employed in CLUS-HMC aims at finding accurate models, since it considers the homogeneity in the values of the target variables and reduces the error on the training data. However, it does not consider the dependencies of the target variables values between related examples and therefore neglects the possible presence of autocorrelation in the training data. To address this issue, we introduced network autocorrelation in the search heuristic and combined it with the variance in a new heuristics.

More formally, the NHMC heuristics is a linear combination of the average autocorrelation measure $A_Y(\cdot)$ (first term) (Algorithm 1 line 8) and the variance reduction $Var(\cdot)$ (second term):

$$Var'(U) = \frac{Var(U) - \delta_{min}}{\delta_{max} - \delta_{min}}, \tag{4}$$

where $Var'(U)$ is the min-max normalization of $Var(U)$ required to keep the values of the linear combination in the unit interval $[0, 1]$, with δ_{max} and δ_{min} being the maximum and the minimum values of $Var(U)$ over all tests.

We point out that the heuristics in NHMC combines information on both the network structure, which affects $A_Y(\cdot)$, and the hierarchical structure of the class, which is embedded in the computation of the distance $d(\cdot, \cdot)$ used in Formula (3) and Algorithm 1 line 8). We also note that the tree structure of the NHMC model makes it possible to consider different effects of the autocorrelation phenomenon at different levels of the tree model, as well as at different levels of the hierarchy (non-stationary autocorrelation). In fact, the effect of the class weights $\omega(c_j)$ is that higher levels of the tree will likely capture the regularities at higher levels of the hierarchy.

In NHMC, the time complexity of selecting a splitting test represents the main cost of the algorithm. The cost is $O(m \cdot (N \cdot logN + N \cdot s) \cdot K) + O(m \cdot d \cdot (N + N \cdot s) \cdot K)$, that is $O(m \cdot N \cdot (logN + d \cdot s) \cdot K)$, where N is the number of examples in the training set, m is the number of descriptive variables, s is the average number of edges for each node in the network and K is the number of classes. This complexity is similar to that of CLUS-HMC, except for the s factor which equals N in the worst case, although such worst-case is unlikely.

The relative influence of the two parts of the linear combination in Algorithm 1, line 8 is determined by a user-defined coefficient α that falls in the interval [0,1]. When $\alpha = 0$, NHMC uses only autocorrelation and when $\alpha = 0.5$, it weights equally variance reduction and autocorrelation. When $\alpha = 1$ NHMC just works as the original CLUS-HMC algorithm.

If autocorrelation is present, examples with high autocorrelation will fall in the same cluster and will have similar values of the response variable. In this way, we are able to keep together connected examples without forcing splits on the network structure (which can result in losing generality of the induced models).

Finally, note that the linear combination that we use in this article (Algorithm 1, line 8) was selected as a results of our previous work on learning from autocorrelated data [29]. The variance and autocorrelation can also be combined in some other way (e.g., as a cross-product). Investigating different ways of combining them is one of the directions for our future work.

5 Empirical Evaluation

In this Section, we present the evaluation of the system NHMC on several datasets related to predicting gene function in yeast. Before we proceed to presenting the empirical results, we provide a description of the datasets used and the experimental settings.

5.1 Data

We use 12 yeast (*Saccharomyces cerevisiae*) datasets as considered by Clare and King [11], but with new and updated class labels [31]. The datasets describe different aspects of the genes in the yeast genome. They include five types of bioinformatics data: sequence statistics, phenotype, secondary structure, homology and expression. The different sources of data highlight different aspects of gene function.

We construct two versions of each dataset. The values of the descriptive attributes are identical in both versions, but the classes are taken from two different classification schemes. In the first version, they are from FUN[2], a scheme for classifying the functions of gene products, developed by MIPS [26]. FUN is a tree-structured class hierarchy; a small part is shown in Fig. 1(a). In the second version of the data sets, the genes are annotated with terms from the Gene Ontology (GO) [1][3], which forms a directed acyclic graph instead of a tree: Each term can have multiple parents (we use GO's "is-a" relationship between terms). Only annotations from the first six GO levels are taken. Note that GO has an order of magnitude more classes than FUN for our datasets. The 24 resulting datasets can be found at the webpage[4].

In addition, we use several protein-protein interaction networks (PPIs) for yeast genes as in Rahmani et al. [23]. In particular, the networks DIP [12], VM [32] and MIPS [21] are used, which contain 51233, 65982 and 38845 interactions among 7716, 2399 and 40374 proteins, respectively. DIP (Database of Interacting Proteins) stores and organizes information on binary protein-protein interactions that are retrieved from individual research articles. VM stores protein-protein interactions that are retrieved from numerous sources, including experimental data, computational prediction methods and public text collections. Finally, MIPS represents interactions between proteins determined on the basis of their signal transduction.

The basic properties of the datasets and of the networks are given in Table 1. Columns 6-8 (% of connected genes) show the percentage of proteins that are covered by each of the PPI networks. On average, only a half of the proteins

[2] `http://www.helmholtz-muenchen.de/en/mips/projects/funcat`
[3] `http://www.geneontology.org`
[4] `http://kt.ijs.si/daniela_stojanova/NHMC/`

Table 1. Basic properties of the datasets and the PPI networks when predicting gene function in yeast. We use 12 yeast (*Saccharomyces cerevisiae*) datasets (as considered by [11]) grouped by their functional (FUN and GO) annotation and 3 different PPI networks (DIP [12], VM [32] and MIPS [21]). In addition, the percentage of connected genes and the percentage of function related genes are presented for each of the networks.

Annotation	Dataset	#Instan-	#Attri-	#Classes	% of connected genes			% of function related genes		
		ces	butes		DIP	VM	MIPS	DIP	VM	MIPS
FUN	seq	3932	476	499	46	43	46	8	11	7
	pheno	1592	67	455	46	34	46	6	6	7
	struc	3838	19629	499	13	43	13	7	10	6
	hom	3848	47035	499	45	43	45	7	10	6
	cellcycle	3757	77	499	72	44	72	2	10	6
	church	3779	550	499	46	44	46	15	9	5
	derisi	2424	63	499	72	69	72	7	10	6
	eisen	3725	79	461	35	31	35	9	10	7
	gasch1	3764	172	499	47	44	47	9	10	6
	gasch2	3779	51	499	47	44	47	7	10	6
	spo	3703	79	499	48	44	48	3	4	3
	exp	3782	550	499	46	44	46	15	9	5
GO	seq	3900	476	4133	46	43	46	15	22	13
	pheno	1587	67	3127	46	34	46	16	18	3
	struc	3822	19629	4132	59	42	59	14	19	12
	hom	3567	47035	4126	48	47	48	14	22	12
	cellcycle	3751	77	4125	47	44	47	17	26	14
	church	3774	550	4131	46	44	46	13	23	13
	derisi	2418	63	3573	73	69	73	11	18	9
	eisen	3719	79	4119	35	31	35	19	25	15
	gasch1	3758	172	4125	47	44	47	19	26	14
	gasch2	3758	51	4131	47	44	47	17	26	14
	spo	3698	79	4119	48	44	48	17	25	14
	exp	3773	550	4131	46	44	46	39	23	13

are known to interact with other proteins. DIP covers the highest percentage of proteins. However, this percentage is not much different from that of the other two networks, especially from MIPS.

In addition, Table 1 shows the percentage of function-relevant interactions. An interaction is considered to be function-relevant if the two proteins involved in the interaction have at least one function in common (with respect to a given hierarchy). As it is possible to see, 6%-23% observed interactions are relevant. However, a closer look at the statistics reveals that the connections are more function-relevant with respect to GO annotations (for all networks) than with respect to FUN annotations. This is expected, as GO contains a much larger number of functions.

5.2 Experimental Setup

In order to evaluate the performance of the proposed NHMC algorithm, we compare it to CLUS-HMC (NHMC works just as CLUS-HMC when $\alpha = 1$). Moreover, we report the results of NHMC with $\alpha = 0$, when it uses only autocorrelation, and with $\alpha = 0.5$, when it equally weights variance reduction and autocorrelation. The performance of the methods is investigated across a range of experimenters.

In the experiments, we deal with several dimensions: different descriptions of the genes, different descriptions of gene functions, and different gene interaction networks. We have 12 different descriptions of the genes from the Clare and King' datasets [11] and 2 class hierarchies (FUN and GO), resulting in 24 datasets with several hundreds of classes each. Furthermore, we use 3 different PPI networks (DIP, VM and MIPS) for each of those.

As suggested by Vens et al. [31], we build models trained on 2/3 of each data set and test on the remaining 1/3. We use the same splitting in order to allow a direct comparison with their work. To prevent over-fitting, we use two pre-pruning methods: minimal number of examples in a leaf (set to 5) and F-test pruning. The latter uses the F-test to check whether the variance reduction is statistically significant at a given level (0.001, 0.005, 0.01, 0.05, 0.1, 0.125).

Following Vens et al. [31], we evaluate the proposed algorithm by using the Average Area Under the Precision-Recall Curve ($\overline{AUPRC}$), i.e., the (weighted) average of the areas under the individual (per class) Precision-Recall (PR) curves, where all weights are set to $1/|C|$, with C the set of classes. The closer the $\overline{AUPRC}$ is to 1.0, the better the model is. In the considered datasets, the positive examples for a given class are rare as compared to the negative ones. A PR curve plots the precision of a classifier as a function of its recall. The points in the PR space are obtained by varying the value for a threshold τ. In the case of NHMC, the threshold ranges from 0 to 1 with a step of 0.02. The evaluation by using PR curves (and the area under them), is the most suitable in this context, because we are more interested in correctly predicting the positive instances (i.e., that a gene has a given function), rather than correctly predicting the negative ones.

5.3 Results

For each of the datasets, the $\overline{AUPRC}$ of CLUS-HMC (which does not consider network information) and NHMC, which uses the DIP, VM and MIPS PPI networks is shown in Table 2. For each algorithm and dataset, both FUN and GO annotations are considered.

The best results are obtained by using CLUS-HMC for FUN annotations and NHMC with $\alpha = 0.5$ for GO annotations (In the latter case, the average good results are mainly due to the results obtained on the cellcycle dataset). This can be explained by the fact that only a half of the genes have at least one connection to other genes in the PPI networks and this is not enough to improve the predictive accuracy of the global predictive HMC model that is constructed using NHMC (over the model constructed by using CLUS-HMC).

NHMC shows competitive results with respect to CLUS-HMC when using FUN annotations. The situation is different in the case of GO annotations, where NHMC outperforms (in almost all the cases) CLUS-HMC. This is mainly due to the the larger percentage of functionally relevant connections (see Table 1)

Table 2. The $\overline{AUPRC}$ of CLUS-HMC ($\alpha = 1$) and NHMC ($\alpha = 0.5$ and $\alpha = 0$) when predicting gene function in yeast. We use 12 yeast (*Saccharomyces cerevisiae*) datasets (as considered by [11]) with 2 different functional annotation schemes (FUN and GO) and 3 networks (DIP, VM and MIPS).

Dataset	FUN annotated datasets							GO annotated datasets						
		DIP		VM		MIPS			DIP		VM		MIPS	
	$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 0$
seq	**.059**	.054	.053	.054	.054	.043	.043	.023	.032	.030	.028	.028	**.033**	**.033**
pheno	**.036**	.035	.028	**.036**	**.036**	.035	.035	.019	.016	.016	**.083**	**.083**	.051	.051
struc	**.030**	.020	.020	.020	.020	**.030**	**.030**	.018	.012	.012	**.066**	**.066**	**.066**	**.066**
homo	**.073**	.020	.023	.020	.020	**.073**	**.073**	.040	.013	.013	.041	.041	**.049**	**.049**
cellcycle	.032	.030	**.037**	.032	.032	.032	.032	.019	**.287**	**.288**	.036	.036	.027	.027
church	**.029**	.020	.020	**.029**	**.029**	.027	.027	.014	.015	.012	**.036**	**.036**	.030	.030
derisi	.027	**.028**	.025	**.028**	**.028**	.027	.027	.017	.015	.017	**.041**	**.041**	.030	.030
eisen	**.047**	.042	.025	.036	.036	.037	.037	.030	.024	.024	**.052**	**.052**	.046	.046
gasch1	.036	.040	.032	**.047**	**.047**	.030	.045	.024	.018	.019	.031	.031	**.040**	**.040**
gasch2	**.034**	**.034**	.027	.029	.029	.028	.030	.020	.021	.021	**.050**	**.050**	.031	.031
spo	**.030**	.029	.025	**.030**	**.030**	.028	.029	.019	.018	.015	.031	.031	**.041**	**.041**
exp	.040	.030	.025	.033	.033	**.045**	**.045**	.023	.017	.016	**.065**	**.065**	.041	.041
Average:	.039	.032	.028	.033	.033	.036	.038	.022	.041	.040	.047	.047	.041	.041

which indicates that DIP; VM and MIPS networks present some form of autocorrelation on the GO labels. Moreover, the results obtained on GO annotated datasets by using NHMC with $\alpha = 0.5$ and $\alpha = 0$ are similar, indicating that the autocorrelation in some cases, dominates the heuristics.

Comparing the results obtained with the use of different PPI networks, it can be seen that there is no clear indication that one network is better than others. In particular, the best network to be included seems to be related to the specific dataset. This means that some networks provide more information for the classification of examples in some datasets than for the classification of other datasets. In this perspective, NHMC can be used in the evaluation of the contribution of single networks for a given classification problem.

We also compare the results of NHMC to the results of recent bio-inspired strategies, which work in the HMC setting, but do not consider network information. These include Artificial Neural Networks (HMC-LMLP), Ant Colony Optimization (hmAnt-Miner), as well as a genetic algorithm for HMC (HMC-GA) [9]. While the first algorithm is a 1-vs-all (it solves several binary classification problems) method based on artificial neural networks trained with the Back-propagation algorithm, the latter two are methods that discover HMC rules.

The algorithms are evaluated on 7 yeast FUN annotated datasets [11] using the same experimental setup as for CLUS-HMC and NHMC. In Table 3, we present the $AU\overline{PRC}$ results obtained by using HMC-GA, HMC-LMLP, hmAnt-Miner and NHMC ($\alpha = 0.5$) on several FUN annotated datasets. NHMC outperforms all other methods by a great margin. An exception is only the church dataset, for which NHMC performs worse than hmAnt-Miner. As in Table 2, CLUS-HMC and NHMC results are comparable (we remind that these results are obtained with the FUN annotation schema and not with the GO annotation schema, where NHMC ouperforms CLUS-HMC of a great margin). Note that $AU\overline{PRC}$ [9] is similar to $\overline{AUPRC}$, but uses weights that consider the number of examples in each class - $AU\overline{PRC}$ is used here to make results easily comparable to results obtained with the other competitive methods.

Table 3. Comparison with other methods.The $\overline{AUPRC}$ of HMC-GA, HMC-LMLP, hmAnt-Miner and NHMC (using $\alpha = 0.5$ and the DIP PPI network), for 7 FUN annotated yeast datasets, as used in Cerri at al. [9].

Dataset	HMC-GA	HMC-LMLP	hmAnt-Miner	CLUS-HMC	NHMC_05
pheno	0.148	0.085	0.162	0.239	0.241
cellcycle	0.150	0.144	0.154	0.172	0.173
church	0.149	0.140	0.168	0.171	0.152
derisi	0.152	0.138	0.161	0.175	0.172
eisen	0.165	0.173	0.180	0.205	0.196
gasch2	0.151	0.132	0.163	0.195	0.186
spo	0.151	0.139	0.174	0.186	0.181

6 Conclusion

In this work, we tackle the problem of multi-label prediction when relationships among the classes (instances may belong to multiple classes and classes are organized into a hierarchy), as well as relationships among the instances (instances may be connected in network data) exist. The use of the latter relationships between the instances introduces autocorrelation and lead to violate the assumption that instances are independently and identically distributed (i.i.d.), which underlines most machine learning algorithms. The main contribution of this work is in the consideration of network autocorrelation in hierarchical multi-label classification (HMC). Specifically, in this work, we presented a definition of network autocorrelation in the HMC setting, introduced an appropriate autocorrelation measure for autocorrelation in such setting and developed the method NHMC for hierarchical multi-label classification from network data.

NHMC has been evaluated in the context of hierarchical gene function prediction in a PPI network context. Given a set of genes with known functions, NHMC learns to predict multiple gene functions when gene classes are hierarchically organized (and, possibly, in the form of DAGs), according to a hierarchical classification scheme (such as the MIPS-FUN and the Gene Ontology) and exploiting a given PPI network (such as DIP, VM and MIPS).

Due to the tree structure of the learned models, NHMC is able to consider the non-stationary effect of autocorrelation i.e., different effects of network autocorrelation at different levels of granularity. However, NHMC does not need the PPI network in the prediction phase, which is beneficial, especially in cases where the prediction needs to be made for new examples (genes) for which connections/interactions to other examples (genes) are not known or still need to be confirmed.

Empirical evidence shows that explicitly taking network autocorrelation into account increases the predictive capability of the models, especially when network interactions express information on the class labels. In future work, we intend to evaluate our approach by using additional datasets and networks, possibly considering new evaluation measures. Moreover, we intend to study a different mechanism to balance variance reduction and autocorrelation. Finally, we intend to evaluate how the sparseness of network information affects predictive capabilities of NHMC.

References

1. Ashburner, M., et al.: Gene ontology: tool for the unification of biology. The gene ontology consortium. Nature Genetics 25, 25–29 (2000)
2. Astikainen, K., Pitkänen, E., Rousu, J., Holm, L., Szedmák, S.: Reaction kernels - structured output prediction approaches for novel enzyme function. Bioinformatics, 48–55 (2010)
3. Barutcuoglu, Z., Schapire, R.E., Troyanskaya, O.G.: Hierarchical multi-label prediction of gene function. Bioinformatics 22(7), 830–836 (2006)
4. Batagelj, V., Mrvar, A.: PAJEK – Program for large network analysis (1998)
5. Bi, W., Kwok, J.T.: Multilabel classification on tree- and dag-structured hierarchies. In: Getoor, L., Scheffer, T. (eds.) ICML, pp. 17–24. Omnipress (2011)
6. Bilgic, M., Getoor, L.: Effective label acquisition for collective classification. In: Proc. 14th ACM SIGKDD Intl. Conf on Knowledge Discovery and Data Mining, pp. 43–51 (2008)
7. Ceci, M.: Hierarchical text categorization in a transductive setting. In: ICDM Workshops, pp. 184–191 (2008)
8. Ceci, M., Malerba, D.: Classifying web documents in a hierarchy of categories: a comprehensive study. J. Intell. Inf. Syst. 28(1), 37–78 (2007)
9. Cerri, R., Barros, R.C., de Carvalho, A.C.P.L.F.: A genetic algorithm for hierarchical multi-label classification. In: Proc. of the 27th Annual ACM Symposium on Applied Computing, SAC 2012, pp. 250–255. ACM (2012)
10. Cesa-Bianchi, N., Gentile, C., Zaniboni, L.: Incremental algorithms for hierarchical classification. J. Mach. Learn. Res. 7, 31–54 (2006)
11. Clare, A., King, R.D.: Predicting gene function in s. cerevisiae. In: Proc. Eur. Conf. on Computational Biology, pp. 42–49 (2003)
12. Deane, C.M., Salwiński, Ł., Xenarios, I., Eisenberg, D.: Protein interactions. Molecular & Cellular Proteomics: MCP 1(5), 349–356 (2002)
13. Doreian, P.: Network Autocorrelation Models: Problems and Prospects. In: Spatial Statistics: Past, Present, and Future. Monograph, vol. 12. Ann Arbor Institute of Mathematical Geography (1990)
14. Gallagher, B., Tong, H., Eliassi-Rad, T., Faloutsos, C.: Using ghost edges for classification in sparsely labeled networks. In: Proc. 14th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining, pp. 256–264 (2008)
15. Jensen, D., Neville, J.: Linkage and autocorrelation cause feature selection bias in relational learning. In: Proc. 9th Intl. Conf. on Machine Learning, pp. 259–266. Morgan Kaufmann (2002)
16. Jensen, D., Neville, J., Gallagher, B.: Why collective inference improves relational classification. In: Proc. 10th Intl. Conf. on Knowledge Discovery and Data Mining, pp. 593–598 (2004)
17. Jiang, X., Nariai, N., Steffen, M., Kasif, S., Kolaczyk, E.: Integration of relational and hierarchical network information for protein function prediction. BMC Bioinformatics 9(1) (2008)
18. Kong, X., Shi, X., Yu, P.S.: Multi-label collective classification. In: SDM, pp. 618–629. SIAM/Omnipress (2011)
19. Macskassy, S., Provost, F.: Classification in networked data: a toolkit and a univariate case study. Machine Learning 8, 935–983 (2007)
20. Macskassy, S.A.: Improving learning in networked data by combining explicit and mined links. In: Proc. 22nd Intl. Conf. on Artificial Intelligence, pp. 590–595 (2007)

21. Mewes, H.W., Heumann, K., Kaps, A., Mayer, K., Pfeiffer, F., Stocker, S., Frishman, D.: Mips: A database for protein sequences and complete genomes. Nucl. Acids Res. 27, 44–48 (1999)
22. Neville, J., Jensen, D.: Relational dependency networks. Journal of Machine Learning Research 8, 653–692 (2007)
23. Rahmani, H., Blockeel, H., Bender, A.: Predicting the functions of proteins in protein-protein interaction networks from global information. Journal of Machine Learning Research 8, 82–97 (2010)
24. Re, M., Valentini, G.: An experimental comparison of hierarchical bayes and true path rule ensembles for protein function prediction. In: El Gayar, N., Kittler, J., Roli, F. (eds.) MCS 2010. LNCS, vol. 5997, pp. 294–303. Springer, Heidelberg (2010)
25. Rousu, J., Saunders, C., Szedmak, S., Shawe-Taylor, J.: Kernel-based learning of hierarchical multilabel classification models. J. Mach. Learn. Res. 7, 1601–1626 (2006)
26. Ruepp, et al.: The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes. Nucleic Acids Research 32(18), 5539–5545 (2004)
27. Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine 3, 93–106 (2008)
28. Steinhaeuser, K., Chawla, N.V., Ganguly, A.R.: Complex networks as a unified framework for descriptive analysis and predictive modeling in climate science. Statistical Analysis and Data Mining 4(5), 497–511 (2011)
29. Stojanova, D., Ceci, M., Appice, A., Džeroski, S.: Network regression with predictive clustering trees. Data Mining and Knowledge Discovery 14 (2012)
30. Valentini, G.: True path rule hierarchical ensembles for genome-wide gene function prediction. IEEE ACM Transactions on Computational Biology and Bioinformatics 8(3), 832–847 (2010)
31. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning 73(2), 185–214 (2008)
32. von Mering, C., Krause, R., Snel, B., Cornell, M., Oliver, S.G., Fields, S., Bork, P.: Comparative assessment of large-scale data sets of protein-protein interactions. Nature 417(6887), 399–403 (2002)

A Density-Based Backward Approach to Isolate Rare Events in Large-Scale Applications

Enikö Székely[1], Pascal Poncelet[1], Florent Masseglia[2], Maguelonne Teisseire[3], and Renaud Cezar[4]

[1] Computer Science Department, University of Montpellier, Montpellier, France
{eniko.szekely,pascal.poncelet}@lirmm.fr
[2] INRIA, Montpellier, France
florent.masseglia@inria.fr
[3] Maison de la Télédetection, Montpellier, France
teisseire@teledetection.fr
[4] Institute for Research in Biotherapy, University Hospital, Montpellier, France
r-cezar@chu-montpellier.fr

Abstract. While significant work in data mining has been dedicated to the detection of single outliers in the data, less research has approached the problem of isolating a group of outliers, i.e. rare events representing micro-clusters of less – or significantly less – than 1% of the whole dataset. This research issue is critical for example in medical applications. The problem is difficult to handle as it lies at the frontier between outlier detection and clustering and distinguishes by a clear challenge to avoid missing true positives. We address this challenge and propose a novel two-stage framework, based on a backward approach, to isolate abnormal groups of events in large datasets. The key of our backward approach is to first identify the core of the dense regions and then gradually augments them based on a density-driven condition. The framework outputs a small subset of the dataset containing both rare events and outliers. We tested our framework on a biomedical application to find micro-clusters of pathological cells. The comparison against two common clustering (DBSCAN) and outlier detection (LOF) algorithms show that our approach is a very efficient alternative to the detection of rare events – generally a recall of 100% and a higher precision, positively correlated wih the size of the rare event – while also providing a $\mathcal{O}(N)$ solution to the existing algorithms dominated by a $\mathcal{O}(N^2)$ complexity.

Keywords: rare events, outlier/anomaly detection, large scale, k-means.

1 Introduction

"An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism" [9]. Similarly, a *rare event* – cluster of outliers [18], clustered anomaly [13,14], anomaly collection [6], micro-cluster [2] – is a group of observations which deviates so much from the other groups of observations as to arouse suspicions that it was generated by a different mechanism.

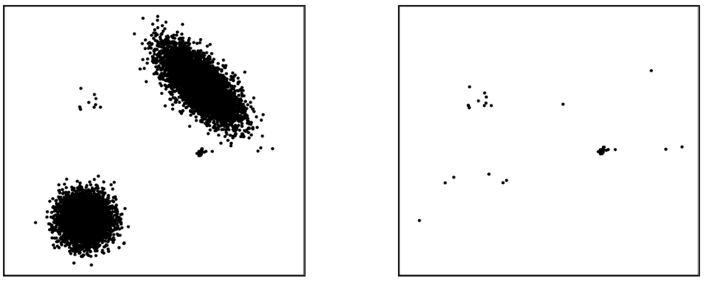

(a) Original Data (b) RARE

Fig. 1. Detection of rare events with RARE on artificially generated data. The dataset contains two normal populations and two rare events: one sparse and global and one dense and local.

The detection of rare events with a high recall, i.e. no false negatives, is intrinsic to those domains where the cost of missing rare events is significantly high. The most representative example is the medical domain where, for example, the cost of missing a pathological group of cells in a blood sample is significantly higher than the cost of classifying a healthy group of cells as pathological, i.e. favouring false positives over false negatives. Disease outbreaks in biosurveillance [19], bursts of clustered attacks [13] or groups of spammers/fraudulent reviewers in social media [6] are other examples of scenarios where the detection of rare events is prevailing over the cost of detecting them.

An anomaly – single or clustered – is an event considered as not normal with respect to a normal behaviour [4]. With any type of anomaly, the open issue is to define normality. For *single outliers*, normality is defined in terms of distance, distribution or neighbourhood similarity with other data instances. For *spatial anomalies*, it is their occurence in a specific region of the space that makes them abnormal. For *collective anomalies*, individual instances are normal but it is their co-occurence that makes them anomalies. For *rare events*, it is their small relative size with respect to other data subpopulations that makes them anomalies. Contrary to collective anomalies, every instance contained in a rare event is an anomaly. We consider an example of rare events detection in Figure 1. The data distribution contains two normal populations of 10,000 points and two rare events: a sparser one of 10 points far from the normal populations, i.e. a *global* anomaly, and a denser one of 20 points close to one of the normal populations, i.e. a *local* anomaly. Figure 1(b) shows the output of our approach, RARE, isolating the rare events from the rest of the data.

Sharing common characteristics with both outliers and clusters, the detection of rare events lies at the frontier between *outlier detection* and *strongly imbalanced/unbalanced clustering*. Both clustering and outlier detection algorithms, by their construction, are generally prone at misclassifying positive examples, i.e. rare events, as negative. Algorithms for unbalanced data have been mainly proposed in supervised scenarios [20] for classification problems in the presence of unbalanced training data where the problem is generally handled using resampling, cost-sensitive or one-class learning methods [5]. In unsupervised scenarios the lack of ground truth information makes the problem even more difficult to handle. One of the main causes is the size balancing effect, as for example in k-means, which tends to reduce the variation in cluster sizes as a trade-off for a better accuracy [21]. In spectral clustering, both RatioCut and Ncut [15] put more emphasis on balancing clusters than on minimizing cut values. Both algorithms propose through the balancing constraints introduced to handle the outlier sensitivity of the initial MinCut solution. On the other hand, outlier/anomaly detection algorithms [1] are very effective at discovering single anomalies. Different approaches (density-based, distance-based, distribution-based) have been proposed in the literature. The most common outlier detection algorithm, LOF [3], Local Outlier Factor, outputs a list of top-k outliers according to an outlierness score obtained by comparing the local density of each point against the local density of the points in its neighbourhood. The performance of LOF depends mainly on the construction of the local neighbourhood (parameter $MinPts$).

In this paper we address this gap between outlier detection and clustering methods. Given our main challenge to avoid false negatives, i.e. avoid missing true positives, we propose a density-based *backward* or bottom-up approach, i.e. going from the most dense regions to the least dense ones. Common outlier detection methods use a forward or top-down approach, i.e. they take the top-k outliers according to an outlierness threshold score. The paper is organized as follows. Section 2 is dedicated to a literature review for finding rare events in large datasets. Section 3 introduces our RARE framework. We first perform a clustering using DenseKMeans, a modified variant of k-means, designed to find and cluster only points that lie in dense regions of the space. In the second step, we gradually augment the dense regions found by DenseKMeans using a density-based sliding region. As soon as the density inside the sliding region fails to fullfill a density condition, we consider to have reached the border of the dense regions. Rare events lie outside these borders. In section 4 experiments on a biomedical data benchmark show that RARE is capable of isolating the rare events with a higher precision than both DBSCAN and LOF. We discuss the advantages and limitations of RARE in Section 5.

2 Related Work

Different approaches [4,7,8,10,13,14,17,22] in the literature have been proposed for the detection of rare events in large datasets. A few techniques approach it as cluster-based anomaly detection [4]: normal instances belong to large and dense clusters, while anomalies either belong to small or sparse clusters. Such methods rely on the output of a clustering algorithm. CBLOF [10] first performs a clustering, using any clustering method, and subsequently separates small from large clusters based on a predefined threshold. Using this threshold, it defines a Cluster-Based Local Outlier Factor (CBLOF) outlierness score by taking into account both the size of the cluster and the distance to the closest cluster center. Overall, the performance of such techniques relies strongly on the choice and quality of the initial clustering.

Employing explicit cluster size constraints is another solution [22] that can be used to handle the detection of rare events in datasets. While the tendency in the literature is to concentrate on balancing clusters, this approach allows to generate a partitioning with different cluster sizes. It can be very helpful when an a priori knowledge on the size of each cluster in the data is known in advance. Still, only a few applications benefit from such a faithful information.

A third approach is to use or adapt single outlier detection algorithms and make them suitable for detecting micro-clusters of outliers. In LOF [3] the detection of outlying clusters depends on the choice of the number of nearest neighbours *MinPts* that define the local neighbourhood. The detection of very small clusters requires a *MinPts* large enough to contain all the points in a cluster, i.e. larger than the size of the cluster. LOCI [17] defines a multi-granularity deviation factor (MDEF) and identifies outliers as those points whose neighbourhood size is significantly different than the neighbourhood size of their neighbours. Similarly to LOF, LOCI relies on an appropriate choice of the neighbourhood size, except that, contrary to LOF, it requires the maximum radius of the neighbourhood as input parameter.

Another different direction is to consider that normal instances belong to a cluster in the data, while outliers do not belong to any cluster [4]. This approach requires the use of methods (DBSCAN [8], SNN-based clustering [7]) that do not force every point to belong to one of the clusters. DBSCAN [8] is the most common density-based clustering algorithm. Its novel notion of *density reachability* allows the detection of clusters of arbitrary sizes ans shapes, but it cannot handle clusters of different densities. Both the run time complexity and memory requirements of the original alorithm are high $\mathcal{O}(N^2)$. Using efficient indexing structures like k-d trees to find the nearest neighbours, the run time complexity can be reduced to $\mathcal{O}(N \log N)$. However such indexing structures are not suitable for high-dimensional data.

A relatively recent concept – *isolation* – was proposed [12,13,14] as an alternative to the concepts of distance and density used in most outlier detection methods. The notion of isolation relies on the property of anomalies of being 'few and different'. The two methods, iForest [12,14] and SCiForest [13], that rely on this concept build in the training phase forests of t binary trees using sub-samplings of the data and compute in an evaluation step an anomaly score based on the path length of each point, defined as the path from the root of the tree to the node. While both methods are effective at discovering global clustered anomalies, i.e. clusters far apart from normal populations, only SCiForest is able to detect local clustered anomalies [14], i.e. clusters close to normal populations (we presented both types of clustered anomalies in our example in Figure 1). However the high complexity of SCiForest in both training and evaluation stages, respectively $\mathcal{O}(t\tau\psi(q\psi + log\psi + \psi))$ and $\mathcal{O}(qNt\psi)$, where ψ is the sampling size for building the iTrees and t the number of trees to build in the training phase, makes it suitable only in the presence of local clustered anomalies.

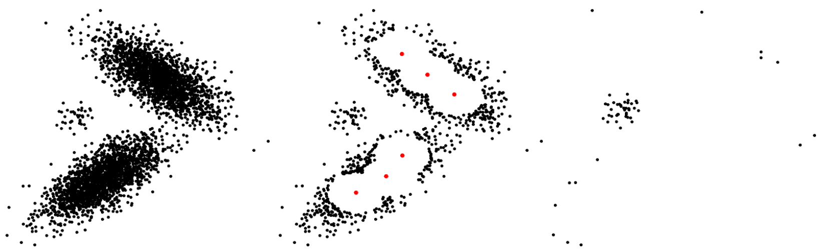


Fig. 2. Illustrative example: a) Original data: the rare event contains 1% of the entire data collection. b) The data subset after eliminating the core of the dense regions with DENSEKMEANS. c) Rare events after DENSESLIDE.

The RARE framework that we propose in this paper proposes: 1) a *backward* approach to the detection of rare events by first identifying the normal/dense regions; 2) an approach designed to avoid false negatives and therefore accepting false positives, favouring recall over precision; 3) a low complexity due to the use of a variant of k-means (linear, scalable); 4) a lower bound density-driven approach in both steps of the framework that allow the detection of rare events.

3 The RARE Framework

We describe in this section our two-stage framework for the detection of rare events in large datasets. Given a dataset $\mathbf{X}$ with N data points, we consider a rare event as a *micro-cluster* of size N_R, where N_R is significantly smaller than the total size of the dataset ($N_R \ll N$).

When expressed in terms of the ratio $\varepsilon = \frac{N_R}{N}$ between the number of points in the rare event and the total number of points in the dataset, the above rare event condition becomes $\varepsilon \ll 1$. Very small values of ε, i.e. $\varepsilon < 10^{-2}$, place the problem of abnormal events detection at the frontier between *outlier detection* and *strongly imbalanced clustering.*

3.1 The Backward Approach: An Illustrative Example

We illustrate the backward approach of RARE by means of an example in Figure 2. We consider a dataset $\mathbf{X}$ with two normal subpopulations and a rare event representing 1% of the whole dataset.

First, we want to identify the core of the dense regions while handling two major issues at this stage: the *scalability* and the *density.* We have no a priori knowledge on the number of subpopulations in the data. To handle the *scalability* issue we choose to cluster the dataset using k-means [16] due to both its linear complexity and parallelization power. The *density* problem is then handled by modifying k-means so that only points that lie in dense regions are clustered. We do this by changing in the re-assignement phase of k-means the way cluster centers are estimated, i.e. only points that lie within a maximum radius around cluster centers contribute to the reestimation of the centers. This radius-limited approach does not force all points to belong to one of the clusters, i.e. some points will be left unclustered. As the actual number of clusters in the dataset is unknown, we use a large initial number of clusters K_I and let each population be modelled using multiple clusters. Figure 2(b) illustrates this first step of the analysis after the convergence of the centers to the core of the dense regions. We use $K_I = 6$ cluster centers in this example and plot the output of DenseKMeans, i.e. the points left unclustered after the first step, $\mathbf{X}_{KEEP}$.

In the second stage (Figure 2(c)) the clusters that belong to the same population, i.e. they are adjacent as will be defined in Section 3.3, are merged to form connected components. In our example each group of 3 clusters forms a connected component. The two components are then gradually augmented, by means of a density-based Gaussian sliding region (DenseSlide), to reach the border of the dense regions. Everything that is outside these borders, $\mathbf{X}_{RARE}$, is considered a rare event. The framework retrieves both true positives, i.e. the rare event, and false positives, i.e. points that lie close to the border of the dense regions or outliers.

3.2 Dense Regions Clustering

The principle behind k-means relies on the minimization of a distance-based objective function that clusters the dataset $\mathbf{X}$ around K cluster centers. But this distance-based approach leaves k-means sensitive to density-related issues and to the presence of outliers and noise. To adress this density problem and cluster only points that lie in dense regions we propose a density-based radius-limited variant of k-means – DenseKMeans – by bringing two modifications to the original algorithm:

$$\min \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in \mathcal{C}_k} \| \mathbf{x}_i - CC_k \|^2 \tag{1}$$
$$\text{s.t. } | \mathcal{C}_k | > N_I$$
$$dist(\mathbf{x}_i, CC_k) < D_{MAX}, \forall \mathbf{x}_i \in \mathcal{C}_k$$

where $\mathcal{C}_k$ denotes the clusters, CC_k their corresponding centers, N_I the minimum cluster density and D_{MAX} the maximum radius around each cluster center that constraints center positioning in the initialization phase and limits the points considered by k-means in the reestimation of cluster centers. Thus the two phases of k-means are modified as in the following:

1. ***initialization***: choose cluster centers iteratively so that each new center is positioned at a minimum of D_{MAX} distance from all the other centers and that each cluster center is assigned at least N_I data points.
2. ***re-assignement***: reestimate cluster centers using only points that are at a maximum of D_{MAX} distance from one of the cluster centers and remove cluster centers that fall below the initial N_I threshold during the re-assignement phase.

Table 1. DenseKMeans

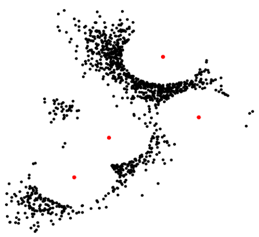
```
Algorithm 1: DenseKMeans
Input: $\mathbf{X} = \{\mathbf{x}_i\}, i = 1..N, \mathbf{x}_i \in \mathbb{R}^D$
        $K_I$ - initial number of clusters
        $N_I$ - minimum number of points (density)
        $D_{MAX}$ - radius
Output: $\mathbf{CC} = \{CC_k\}, k = 1..K_F$ - final cluster centers
        $\mathbf{X}_{KEEP}$ - the subset of points left unclustered
        $\mathbf{X}_{RMV}$ - the subset of points clustered
Initialization:
1': Choose cluster centers CC iteratively so that they are further than $D_{MAX}$ one from each other:
        $\|CC_k, CC_l\|_2 > D_{MAX}, \forall k, l = 1..K_I$
2': Check the density condition: $| \mathcal{C}_k | > N_I$
3': Repeat steps 1' and 2' until convergence: all $K_I$ centers are assigned at least $N_I$ points.
DenseKMeans:
1": Select all points $\mathbf{X}_{KEEP}$ that are further than $D_{MAX}$ from all centers:
        $min(\mathbf{x}_i, CC_k) > D_{MAX}$
2": Reestimate cluster centers using $\mathbf{X}_{RMV} = \mathbf{X} \setminus \mathbf{X}_{KEEP}$
3": If a cluster center falls under the initial density threshold ($| \mathcal{C}_k | < N_I$) remove it.
4": Repeat steps 1"-3" until convergence: a maximum number of iterations is reached or centers do not change significantly.
```

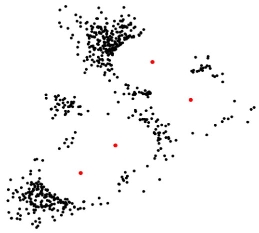
DenseKMeans is summarized in Table 1. The reestimation of cluster centers using only points that are at a maximum of D_{MAX} distance from one of the cluster centers eliminates k-means' sensitivity to outliers – in our case to rare events – as long as the radius D_{MAX} is smaller than the distance to outliers. Moreover clusters $\mathcal{C}_k$ that are not dense enough, $| \mathcal{C}_k | < N_I$, are discarded in the re-assignement phase.

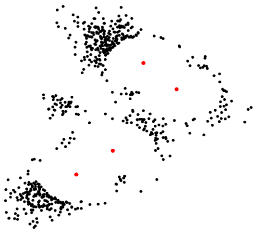
These two modifications allow to restrict the region of the space considered by k-means to only dense regions and iteratively move cluster centers towards the core of the dense regions. Figure 3 illustrates a few examples with different parameter combinations D_{MAX} vs. K: 1) $D_{MAX} = 1.4$, $K_I = 4$ (Figure 3(a, b, c)); 2) $D_{MAX} = 1.2$, $K_I = 6$ (Figure 3(d, e, f);) 3) $D_{MAX} = 1$, $K_I = 8$ (Figure 3(g, h, i)). The output of this first stage of the algorithm divides the original dataset into two disjoint subsets $\mathbf{X} = \mathbf{X}_{RMV} \cup \mathbf{X}_{KEEP}$: 1) $\mathbf{X}_{RMV}$ = points falling within a maximum of D_{MAX} distance from the final cluster centers, 2) $\mathbf{X}_{KEEP}$ = points falling outside the region defined by the maximum D_{MAX} distance from the final cluster centers. Using this approach, only points that are in dense regions are clustered.

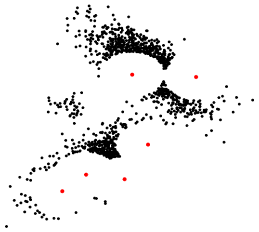
3.3 Dense Regions Augmentation

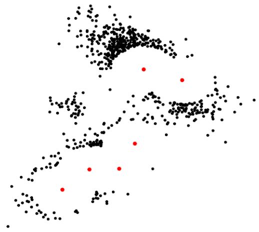
DenseKMeans identifies the core of the dense regions using an initial number of clusters K_I larger than the actual number of clusters/data subpopulations. The radius-limited approach of DenseKMeans allows to define the *cluster adjacency* property as in the following:

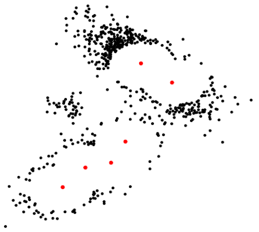

(a) $\mathbf{X}$ (initialization)

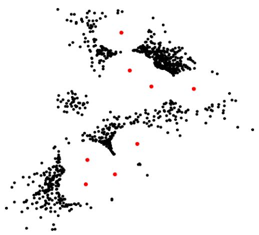

(b) $\mathbf{X}_{KEEP}$ (intermediary step)

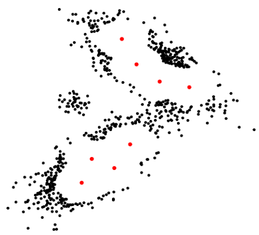

(c) $\mathbf{X}_{KEEP}$ (final)


(d) $\mathbf{X}$ (initialization)


(e) $\mathbf{X}_{KEEP}$ (intermediary step)


(f) $\mathbf{X}_{KEEP}$ (final)


(g) $\mathbf{X}$ (initialization)


(h) $\mathbf{X}_{KEEP}$ (intermediary step)

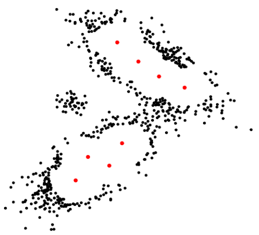

(i) $\mathbf{X}_{KEEP}$ (final)

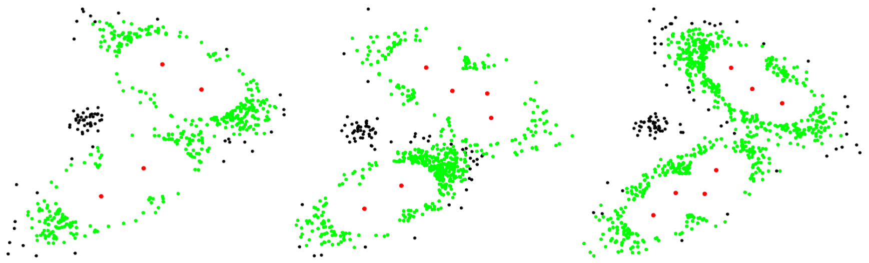
Fig. 3. Varying D_{MAX} and K_I in DenseKMeans considering the original data from Figure 2: (a,b,c) $D_{MAX} = 1.4$, $K_I = 4$; (d,e,f) $D_{MAX} = 1.2$, $K_I = 6$; (g,h,i) $D_{MAX} = 1$, $K_I = 8$. Red points represent cluster centers. The initial, intermediary and final step for each case illustrate the convergence of cluster centers towards the core of the dense regions, eliminating the sensitivity of the original k-means to outliers.

Definition 1. *Two clusters defined by centers CC_k and CC_l and maximum radius D_{MAX} are adjacent if they are overlapping, i.e. the Euclidean distance between the centers CC_k and CC_l is less than $2 \times D_{MAX}$:*

$$\|CC_k, CC_l\|_2 < 2 \times D_{MAX}$$

Among the final K_F dense clusters found by DenseKMeans, adjacent clusters are merged to build connected components and provide a more faithful representation of the real data subpopulations.

A spherical model like the one used by k-means and DenseKMeans considers that the intrinsic dimensionality of the data is equal to the original dimensionality. However in real scenarios the intrinsic dimensionality of the data - especially locally, i.e. one data subpopulation/cluster - is rarely equal to the original dimensionality [11]. To address this challenge, we treat the output of the spherical model by means of a model that is better adapted to handle the intrinsic dimensionality of the data. The most common is the Gaussian model. In the first step of the analysis, the spherical approach was preferred due to the scalability advantage of k-means. The use of the Gaussian mixture model in the first step would have required the estimation of $K(D^2+D+1)$ parameters for every value of K – as K is not known in advance. Even if parsimonius models, e.g. diagonal, can replace the full Gaussian model, the challenge to detect rare events is too sensitive and requires the use of a full model.



(a) $D_{MAX} = 1.4$, $K_I = 4$ (b) $D_{MAX} = 1.2$, $K_I = 6$ (c) $D_{MAX} = 1$, $K_I = 8$

Fig. 4. Points in green are eliminated through DenseSlide. The same combinations of D_{MAX} and K_I as in Figure 3 are used. c) Only 7 out of 8 clusters are left, one was eliminated because it did not fullfill the density condition ($| \mathcal{C}_k | < N_I$) in DenseKMeans.

The subset $\mathbf{X}_{RMV}$ allows to quickly estimate both the means μ_j and covariance matrices Σ_j of the core dense regions defined by the connected components. These dense regions are augmented using a sliding region S_R defined based on the Mahalanobis distance D_M and an increase parameter ϵ_S. The sliding regions approach the border of the dense regions gradually and the process is repeated as long as a density condition is fullfilled, $nbPoints(S_R) > N_S$, i.e. the number of points inside the sliding region is larger than a predefined threshold N_S. When the density inside the sliding region drops below this threshold, we consider to have reached the border of the dense regions. The algorithm for dense regions augmentation, DenseSlide, is summarized in Table 2 and a few examples for various combinations of parameters D_{MAX} and K_I are shown in Figure 4. The parameters for DenseSlide were $\epsilon_S = 0.1$ and $N_S = 10$. The output of the algorithm returns the subset $\mathbf{X}_{RARE}$ of positive examples.

Table 2. DenseSlide

```
Algorithm 2: DenseSlide
Input: $\mathbf{X}_{KEEP}$, $\mathbf{X}_{RMV}$, CC - output of DenseKMeans
       $\epsilon_S$ - increase parameter for the sliding region
       $N_S$ - number of points in the sliding region
Output: $\mathbf{X}_{RARE}$ - output of RARE
Connected components:
1': Build the graph $G = (\mathbf{CC}, E)$ using the cluster adjacency property.
2': Find connected components $G_j$ in $G$.
3': Use $\mathbf{X}_{RMV}$ to model $G_j$ as $\mathcal{N}(\mu_j, \Sigma_j)$.
Sliding Region:
1": Initialize $\mathbf{X}_{RARE}$=$\mathbf{X}_{KEEP}$.
2": For each $G_j$ compute the Mahalanobis distance:
       $D_M^j = \sqrt{(\mathbf{X}_{RARE} - \mu_j)^T \Sigma_j^{-1} (\mathbf{X}_{RARE} - \mu_j)}$
3": Eliminate points from $\mathbf{X}_{RARE}$ that are closer to one of the component centers than the
    farthest point from $\mathbf{X}_{RMV}$: $D_M^j(x_i) > D_{max}^j$.
4": Create a moving sliding region $S_R(D_{max}^j, \epsilon_S)$ around each component $\mathcal{N}(\mu_j, \Sigma_j)$.
5": Eliminate points from $\mathbf{X}_{RARE}$ inside $S_R$.
6": Repeat steps 4" and 5" as long as the density condition is respected: $nbPoints(S_R) > N_S$.
```

4 Experimental Results

In this section we test RARE on a large-scale biomedical application in a diagnosis purpose, to isolate pathological group of cells in flow cytometry. We perform experiments on multiple data sets with varying sizes of the rare event. A practical analysis of the influence of parameter values is also performed on the benchmark data. Finally we compare RARE against both clustering – DBSCAN – and outlier detection – LOF – algorithms[1]. We experiment with various parameter values to illustrate the behaviour of each of the above methods.

We use Precision and Recall to evaluate the performance of the algorithms. RARE is an unsupervised method and ground-truth information on positive and negative examples is used only in the evaluation phase. Given our main challenge to avoid missing true positives, it is Recall that becomes the most important evaluation measure in this scenario.

$$\mathrm{P} = \frac{TP}{TP+FP} = \frac{TP}{|\mathbf{X}_{RARE}|}, \qquad \mathrm{R} = \frac{TP}{TP+FN} = \frac{TP}{N_{RS}} \tag{2}$$

where $|\mathbf{X}_{RARE}|$ = the number of data points retrieved by RARE and N_{RS} = the number of positives in the data, i.e. the size of the rare event.

4.1 A Real Case: Flow Cytometry

In flow cytometry each cell is characterised by fluorescence levels in response to cell markers, i.e. attributes. Nowadays flow cytometers can count up to tens of

[1] We used the ELKI implementation available at: `http://elki.dbs.ifi.lmu.de/`.

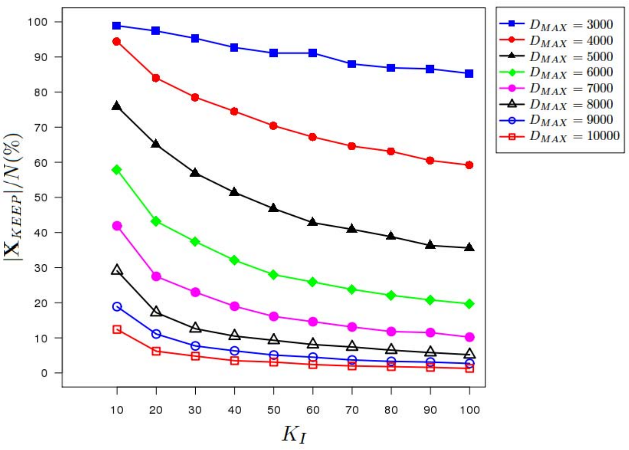


Fig. 5. Initialization of D_{MAX} and K_I for the flow cytometry dataset

millions of cells representing normal cell populations found in any healthy patient, such as lymphocytes or monocytes. In patients presenting a blood pathology, the blood samples also contain micro-clusters of cells with abnormal signatures, i.e. abnormal combinations of cell marker fluorescence levels. The human detection of these rare events is performed visually by sequentially inspecting two-dimensional spaces, i.e. combinations of two markers.

Experiment 1. D_{MAX} **and** K_I. We first estimate the percentage of data covered by DenseKMeans in the first step of the algorithm for various values of the parameters D_{MAX} and K_I (Figure 5). We fixed $N_I = \frac{N}{100 \times K_I}$ because we know that the rare event is significantly smaller that the total size of the dataset. Those combinations of values for D_{MAX} and K_I – closely related – covering approximately $80-90\%$ of the dataset in DenseKMeans ($\mathbf{X}_{RMV}$) generally led to very good final results in the experiments. This is due to the fact that the rare events represent significantly less than the rest of $10-20\%$ of the whole dataset, allowing in the meantime the detection of the core dense regions by DenseKMeans.

Throughout our evaluation, we experimented with different values of the parameters and observed that the choice of the parameter values was consistent across different datasets for a given application.

Experiment 2: Varying N_R. We now wish to test the performance of RARE for varying levels of unbalancedness. In this purpose we will keep the total size of the dataset fixed and vary the size of the rare event - which is an indicator of the phase of the pathology. On the biological side, this experiment was performed by injecting grown cells from a blood pathology into a cell sampling of a healthy patient. The size of the rare population injected was of $\{5, 10, 20, 50, 100, 500\}$.

Due to flow cytometers machine error, a difference appears between the number of injected cells and the actual size N_{RS} of the rare cell population found in the blood samples, i.e. positive examples (corresponding to a pathology signature in flow cytometry). The whole dataset contained $N = N_H + N_{RS}$ cells, where $N_H \approx 700.000$ cells. In this experiment the free parameters D_{MAX} and K_I in DenseKMeans were chosen to guarantee the ratio $\frac{|\mathbf{X}_{KEEP}|}{N} \approx 10 - 20\%$ across the different blood samples (as discussed in Experiment 1). Here we choose $D_{MAX} = 8000$ and $K_I = 40$, but other value combinations that respect the above ratio are also valid (as will be seen in Experiment 3). The parameters for DenseSlide were chosen: $\epsilon_S = 0.1$ and $N_S = 10$.

Table 3. RARE on three samples for each of the varying $N_R = \{5, 10, 20, 50, 100, 500\}$

N_R	N	$\lvert\mathbf{X}_{RARE}\rvert$	TP	FP	P	R	N_{RS}
0	151,388	64	**5**	59	7.8%	100%	**5**
5	646,149	42	**4**	38	9.5%	100%	**4**
10	780,988	54	**13**	39	24%	92.8%	**14**
20	757,234	70	**17**	53	24.2%	100%	**17**
50	752,987	65	**30**	35	46.1%	96.7%	**31**
100	760,842	132	**80**	52	60.6%	97.5%	**82**
500	718,743	415	**358**	57	86.2%	99.7%	**359**
0	696,465	102	**14**	88	13.7%	100%	**14**
5	731,576	98	**9**	89	9.1%	75%	**12**
10	720,945	114	**14**	100	12.2%	100%	**14**
20	484,285	129	**25**	104	19.3%	96.1%	**26**
50	630,341	40	**35**	5	87.5%	97.2%	**36**
100	676,745	142	**69**	77	48.5%	98.5%	**70**
500	516,981	541	**366**	175	67.6%	98.6%	**371**
0	671,582	94	**8**	86	8.5%	100%	**8**
5	707,535	100	**7**	93	7%	100%	**7**
10	714,081	135	**13**	122	9.6%	100%	**13**
20	621,155	155	**11**	144	7%	100%	**11**
50	599,851	144	**26**	118	18%	100%	**26**
100	711,801	204	**84**	120	41.1%	100%	**84**
500	993,671	552	**312**	240	56.5%	100%	**312**

The results in Table 3 show an excellent performance for RARE which finds almost all positive examples, i.e. true positives TP (column 3), among the positive examples N_{RS} found with the signature provided by domain experts (column 5). The size of the false positives FP returned by RARE (column 4) depends mainly on the size and structure of the original dataset, i.e. FP remains relatively constant with increasing TP. We also observe that the recall is relatively high and the precision increases with the size of the rare event.

Experiment 3. Comparison with DBSCAN and LOF. A comparison of the parameters required by the three methods is presented in Table 4. While LOF requires only one parameter – $MinPts$ – in the construction phase, DBSCAN and RARE both require two parameters, thus adding more flexibility but also more complexity to the model. Both RARE and LOF require a stopping criteria while DBSCAN considers all points left unclustered as noise. Rare events will often fall in the noise category with DBSCAN (as shown in the next experiment). RARE uses two parameters – ϵ_S and N_S, the growing rate of the sliding region and the minimal density (ϵ_S is generally fixed to either 10^{-1} or 10^{-2}) – to define the stopping criteria. Their influence is equivalent to the cutting threshold in LOF, but it is the approach that is different: LOF has a top-down approach while RARE has a bottom-up approach. The bottom-up approach is preferred in scenarios where avoiding false negatives is the priority.

Table 4. Parameters in RARE, DBSCAN and LOF

Method	Model parameters	Stopping criteria	Approach
RARE	(D_{MAX}, K_I)	(ϵ_S, N_S)	Bottom-up (backward)
DBSCAN	$(\epsilon, MinPts)$	–	Bottom-up
LOF	$MinPts$	$Threshold$ or top-k	Top-down (forward)

In Table 5 we analysed a data sample chosen at random from the second experiment with a medium rare event (752987 samples and 31 positive examples) using various parameter values for the three methods. We compute the number of true positives (TP) and false positives (FP) retrieved by the algorithms. Both RARE and DBSCAN have a high recall (generally 100%) while RARE has a significantly higher precision than DBSCAN. In DBSCAN for most parameter values the rare event is left unclustered and belongs to the subset classified as noise[2] – except in the two cases where a fraction of the rare event clusters separately in a small cluster (14 and 25 points). While DBSCAN requires the $MinPts$ parameter to be lower than the size of the rare event for a relatively good performance, LOF on the contrary requires the $MinPts$ parameter higher than the size of the rare event, i.e. this is necessary for the detection of micro-clusters in LOF. While DBSCAN requires no stopping criteria, in LOF we need to choose either the cutting threshold value or the number of outliers. We use here two cutting threshold values for each value of $MinPts$ in LOF and indicate the number of false positives in each case. The two values were chosen so that the vast majority of the rare event has an LOF outlierness score in the range bounded by the two values.

[2] Here $TP + FP$ equals the size of the noise subset in DBSCAN.

Table 5. Comparison between RARE, DBSCAN and LOF. The parameter values in the second column correspond to the respective parameters of each method from the first column.

Method	Parameters	TP	FP	P	R
RARE($D_{MAX}, K_I, \epsilon_S, N_S$)	(6000, 80, 0.1, 10)	31	193	13.8%	100%
	(6000, 100, 0.1, 10)	31	48	39.2%	100%
	(7000, 40, 0.1, 10)	31	43	41.8%	100%
	(7000, 60, 0.1, 10)	31	60	34%	100%
	(7000, 80, 0.1, 10)	31	57	35.2%	100%
	(7000, 100, 0.1, 10)	30	40	42.8%	96.7%
	(8000, 20, 0.1, 10)	31	184	14.4%	100%
	(8000, 40, 0.1, 10)	31	60	34%	100%
	(8000, 60, 0.1, 10)	31	22	58.4%	100%
	(9000, 10, 0.1, 10)	31	284	9.8%	100%
	(9000, 30, 0.1, 10)	31	48	39.2%	100%
	(9000, 50, 0.1, 10)	31	35	46.9%	100%
	(10000, 10, 0.1, 10)	31	51	37.8%	100%
	(10000, 30, 0.1, 10)	31	35	46.9%	100%
DBSCAN($\epsilon, MinPts$)	(5000, 10)	31	1286	2.3%	100%
	(5000, 20)	31	1998	1.5%	100%
	(5000, 30)	31	2703	1.1%	100%
	(6000, 10)	31	457(14)	6.1%	100%
	(6000, 20)	31	699	4.2%	100%
	(6000, 30)	31	934	3.2%	100%
	(7000, 10)	31	197(25)	12.2%	100%
	(7000, 20)	31	331	8.5%	100%
	(7000, 30)	31	396	7.2%	100%
LOF($MinPts, Threshold$)	(30, 1)	31	589039	5×10^{-3}%	100%
	(30, 1.1)	3	132890	2×10^{-3}%	9.6%
	(50, 1.5)	31	2133	1.4%	100%
	(50, 1.6)	8	945	8×10^{-3}%	25%
	(100, 2)	31	230	11.8%	100%
	(100, 2.5)	3	54	5.2%	9.6%
	(150, 2.1)	31	206	13%	100%
	(150, 2.7)	3	43	6.5%	9.6%

5 Discussion and Conclusion

We proposed in this paper a two-stage framework to isolate rare events in large datasets. The size of these events makes their detection difficult by both clustering and outlier detection algorithms as both tend to missclasify true positives as false negatives. We have shown that RARE has a good performance and also the advantage of the linear complexity, largely dominated by the complexity of k-means and low memory requirements $\mathcal{O}(NK_I)$. The new variant of k-means was proposed to handle the scalability and density issues in this type of problems and the sliding region was designed in a backward/bottom-up approach to avoid false negatives. Overall, the RARE framework targets applications where recall prevails over precision. We did not approach here complexity improvements. Both DBSCAN and LOF have a $\mathcal{O}(N^2)$ memory requirement and runtime complexity – that can be improved to $\mathcal{O}(N \log N)$ using indexing structures such as k-d trees for low-dimensional data. In its current stage RARE has a $\mathcal{O}(N)$ complexity and DenseKMeans is easily parallelizable – it is the most time consuming in RARE. We consider these complexity improvements as a next step for future work.

Acknowledgements. This work was supported by the Labex NUMEV at the University of Montpellier, France. The authors would like to thank Jean-Pierre Vendrell from the Institute for Research in Biotherapy in Montpellier, France for enabling the access to the data for the experimental part and Dino Ienco, Andre Mas, Pierre Pudlo and Sandra Bringay for valuable discussions.

References

1. Aggarwal, C.: Outlier analysis. Springer (2013)
2. Bae, D.-H., Jeong, S., Kim, S.-W., Lee, M.: Outlier detection using centrality and center-proximity. In: Proceedings of CIKM (2012)
3. Breunig, M., Kriegel, H.-P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: Proceedings of ACM SIGMOD (2000)
4. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Computing Surveys 41 (2009)
5. Chawla, N.V., Japkowich, N., Kolcz, A.: Editorial: special issue on learning from imbalanced data. SIGKDD Explorations 6 (2004)
6. Dai, H., Zhu, F., Lim, E.-P., Pang, H.: Mining coherent anomaly collections on web data. In: Proceedings of CIKM (2012)
7. Ertoz, L., Steinbach, M., Kumar, V.: Finding clusters of different sizes, shapes and densities in noisy, high-dimensional data. In: Proceedings of SDM (2003)
8. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of ACM SIGKDD (1996)
9. Hawkins, D.: Identification of outliers. Chapman and Hall (1980)
10. He, Z., Xu, X., Deng, S.: Discovering cluster-based local outliers. Pattern Recognition Letters 24 (2003)
11. Levina, E., Bickel, P.J.: Maximum likelihood estimation of intrinsic dimension. In: Advances in Neural Information Processing Systems, vol. 17 (2005)
12. Liu, F.T., Ting, K.M., Zhou, Z.-H.: Isolation forest. In: Proceedings of ICDM (2008)
13. Liu, F.T., Ting, K.M., Zhou, Z.-H.: On detecting clustered anomalies using SCiForest. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part II. LNCS (LNAI), vol. 6322, pp. 274–290. Springer, Heidelberg (2010)
14. Liu, F.T., Ting, K.M., Zhou, Z.-H.: Isolation-based anomaly detection. ACM Transactions on Knowledge Discovery from Data 6 (2012)
15. von Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing 17 (2007)
16. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability (1967)

17. Papadimitriou, S., Kitagawa, H., Gribbons, P.B., Faloutsos, C.: Loci: Fast outlier detection using the local correlation integral. In: Proceedings of ICDE (2003)
18. Rocke, D.M., Woodruff, D.L.: Identification of outliers in multivariate data. Journal of the American Statistical Association (1996)
19. Shmueli, G., Burkom, H.: Statistical challenges facing early outbreak detection in biosurveillance. Technometrics, Special Issue on Anomaly Detection (2010)
20. Tang, Y., Zhang, Y.-Q., Chawla, N.V., Krasser, S.: Svms for highly imbalanced classification. IEEE Transactions on Systems, Man and Cybernetics 39 (2009)
21. Xiong, H., Wu, J., Chen, J.: K-means clustering versus validation measures: a data distribution perspective. In: Proceedings of SIGKDD (2006)
22. Zhu, S., Wang, D., Li, T.: Data clustering with size constraints. Knowledge-Based Systems 23 (2010)

Inductive Process Modeling of Rab5-Rab7 Conversion in Endocytosis

Jovan Tanevski[1,2], Ljupčo Todorovski[3],
Yannis Kalaidzidis[4], and Sašo Džeroski[1,2,5]

[1] Department of Knowledge Technologies, Jožef Stefan Institute,
Jamova cesta 39, 1000 Ljubljana, Slovenia
[2] Jožef Stefan International Postgraduate School,
Jamova cesta 39, 1000 Ljubljana, Slovenia
{jovan.tanevski,saso.dzeroski}@ijs.si
[3] Faculty of Administration, University of Ljubljana,
Gosarjeva ulica 5, 1000 Ljubljana, Slovenia
ljupco.todorovski@fu.uni-lj.si
[4] Max Planck Institute of Molecular Cell Biology and Genetics,
Pfotenhauer Straße 108, 01308 Dresden, Germany
kalaidzi@mpi-cbg.de
[5] Centre of Excellence for Integrated Approaches in Chemistry and Biology
of Proteins, Jamova cesta 39, 1000 Ljubljana, Slovenia

Abstract. Inductive process modeling (IPM) is an approach to equation discovery that can be used to induce comprehensible models of dynamical systems from observed data and domain knowledge. We apply IPM to the task of modeling the conversion of Rab5 domain proteins to Rab7 domain proteins, a key process in endocytosis. Endocytosis, and in particular its specific form phagocytosis, is a major mechanism of the immune system, used to remove pathogens. We first introduce a formal representation of the domain knowledge for modeling this process. We then present the design of the IPM experiments using the domain knowledge and measured data and the results obtained from these experiments. We finally compare our results with results already published in the literature.

1 Introduction

Equation discovery is an area of machine learning concerned with the discovery of scientific laws and models in the form of equations from observations [8,5]. Inductive process modeling (IPM) is an approach to equation discovery [1,4]. IPM combines domain specific knowledge, describing constituent entities and interactions between them (processes) in a formal language, and observations in the form of time-series data in order to induce explanatory models of dynamic systems. Process models correspond to (ordinary) differential equation (ODE) models. The IPM approach performs both identification of the structure and estimation of the parameter values of ODEs by using the provided knowledge and data. In this paper, we apply the IPM approach to modeling a dynamical system in the domain of systems biology.

A recent paper by del Conte-Zerial et al. [2] presents a mathematical model of the dynamics of a part of the immune response, i.e., the process of endocytosis. More specifically, they model the conversion of Rab5 domain proteins to Rab7 domain proteins, which occurs during endosome maturation. The authors propose a Rab5-Rab7 conversion model based on an extensive comparative analysis of a number of model structures and different kinetic laws for individual reactions. In a follow-up paper, Tashkova et al. [9] address the task of parameter estimation in a single Rab5-Rab7 conversion model structure and explore the use of several parameter estimation methods under a range of observation scenarios involving data of different completeness and accuracy of interpretation.

In this paper, we generalize the results of Tashkova et al. [9] by integrating the proposed parameter estimation methods within IPM, i.e., within the recently developed IPM tool ProBMoT [10]. The IPM paradigm allows us to consider a whole range of candidate structures considered in del Conte-Zerial et al. [2] in an automated manner. We conjecture that IPM can reproduce their results, i.e. select the model structure obtained by following the results of their manual comparative analysis.

We first give an overview of the process of automated modeling using ProBMoT as a specific IPM approach (Section 2). We then formulate the domain knowledge (Section 3) for the modeling task at hand. We next run ProBMoT on this domain knowledge and the data used by del Conte-Zerial et al. [2] and Tashkova et al. [9]. We present the obtained results and compare them with the results of the manual modeling experiment in Section 4. We give a summary of our work and propose directions for further work in Section 5.

2 Inductive Process Modeling and ProBMoT

Inductive Process Modeling (IPM) is a data and knowledge driven machine learning approach to inducing explanatory process-based models of dynamic systems in the form of equations. One process-based model describes a dynamic system as a set of entities and processes which govern the interactions between the entities. Each process model takes the form of a differential or algebraic equation, which describes one specific interaction between entities. Taken together, these processes and models thereof are combined into a system of differential equations that can be used to simulate the dynamic behavior of the observed system.

The ProBMoT [10] tool is a recently developed IPM tool for automated modeling of dynamical systems. A graphical description of the process of automated modeling using ProBMoT is presented in Figure 1. ProBMoT takes as input time-series data about the dynamic behavior of the observed system. It also takes as input modeling knowledge about the studied domain, represented as a library of template model components - entities and processes. Finally, it takes as input an incompletely specified, partial model, described by components from the library.

The library of domain knowledge is a collection of model fragments that can be used in the process of generation of candidate model structures. In the formalism of ProBMoT, the library consists of template entities and processes. The templates give general description of the properties of the entities and the form of the interactions between them.

The conceptual model is a high level general description of the modeled system. It determines which parts of the library will be taken into account in the process of generation of candidate model structures and in which manner.

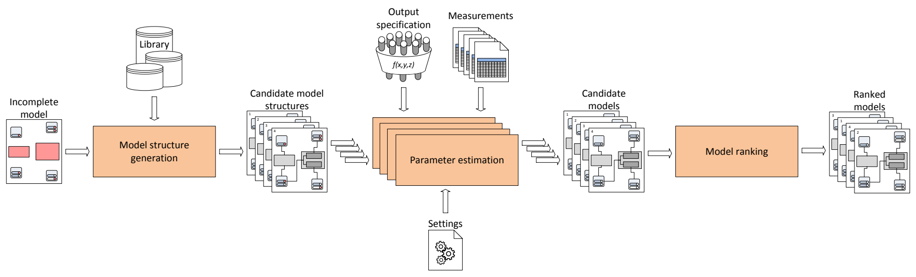


Fig. 1. The process of automated modeling with ProBMoT

Given the library of domain knowledge, ProBMoT performs exhaustive search of the space of candidate model structures by enumerating all possible structures stemming from the conceptual model and the library. Each resulting candidate model structure at this point has the form of a specific set of differential equations with known structure and unknown parameter values. These unknown parameter values are estimated in the next step of the process.

Parameter estimation is performed for each candidate model structure to obtain a set of point estimates of each unknown parameter that most adequately explains the measured behaviour. The parameter estimation task is guided by an objective function which takes into account the error between the measured data and the model simulation resulting from a given set of candidate parameter values for the model. Since all variable properties of the entities that constitute the model might not be observed and present in the measurements data, an output specification, which describes the mapping between the measured and simulated values, can be provided.

ProBMoT uses the jMetal framework [3] for parameter estimation. The jMetal framework contains a collection of state-of-the-art single and multi-objective optimization algorithms. ProBMoT allows the selection of suitable optimization algorithm for the task at hand. The SUNDIALS suite [7] is used for the simulation of the models.

The output of the parameter estimation task is a complete candidate model. After the parameter values for all candidate model structures have been estimated, the resulting candidate models are ranked by their error or by other defined criterion. Finally, the ordered set of candidate models is the output of the process of automated modeling with ProBMoT.

3 Process-Based Knowledge for Modeling Endocytosis

In the library of domain knowledge for modeling endocytosis, entities correspond to protein domains, while processes refer to interactions among the protein domains. In order to develop the library of knowledge for this area of study we used the modular formulation of the Rab5-Rab7 conversion model from del Conte-Zerial et al. [2]. The latter can be represented by the system of ordinary differential equations (ODEs) given below (Equation 1).

$$\begin{aligned}
\frac{dr_5}{dt} &= K_1 - (k_1 + GEF_5(R_5, R_7))r_5 + GAP_5(R_5, R_7)R_5 \\
\frac{dR_5}{dt} &= GEF_5(R_5, R_7)r_5 - GAP_5(R_5, R_7)R_5 \\
\frac{dr_7}{dt} &= K_2 - (k_2 + GEF_7(R_5, R_7))r_7 + GAP_7(R_5, R_7)R_7 \\
\frac{dR_7}{dt} &= GEF_7(R_5, R_7)r_7 - GAP_7(R_5, R_7)R_7
\end{aligned} \tag{1}$$

In Equation 1, the variables r_5 and r_7 represent the concentrations of GDP-bound (passive state) Rab5 and Rab7 domain proteins, while R_5 and R_7 represent the concentrations of GTP-bound (active state) proteins. The parameters K_i and k_i represent GDP Dissociation Inhibitor (GDI) association rates and GDI dissociation fluxes. The Rab5-Rab7 interactions labelled with GEF represent activating reactions which catalyse the GDP/GTP exchange by guanine nucleotide exchange factors, while the GAP interactions represent reactions which catalyse the GTP hydrolysis by means of GTPase-activating proteins. Both GEF and GAP interactions are functions of the GTP-bound state concentrations of Rab5 and Rab7.

Del Conte-Zerial et al. [2] considered different functional forms for modeling the GEF and GAP interactions. The combinations of the different functional forms result in different model structures. Figure 2 provides a graphical representation of the general modular model structure considered by del Conte-Zerial et al. [2].

The dashed lines represent different individual optional interactions between the Rab5 and Rab7 protein domains, while the solid lines represent non-optional (mandatory) interactions. The green arrow lines correspond to activation reactions which increase the conversion from GDP-bound (passive) to GTP-bound (active) states. The red arrow lines correspond to the increase of hydrolysis from GTP-bound to GDP-bound states. Finally, the blue line with a diamond ending represents an inhibitory reaction which decreases the conversion from GDP-bound to GTP-bound states of the corresponding Rab domain proteins.

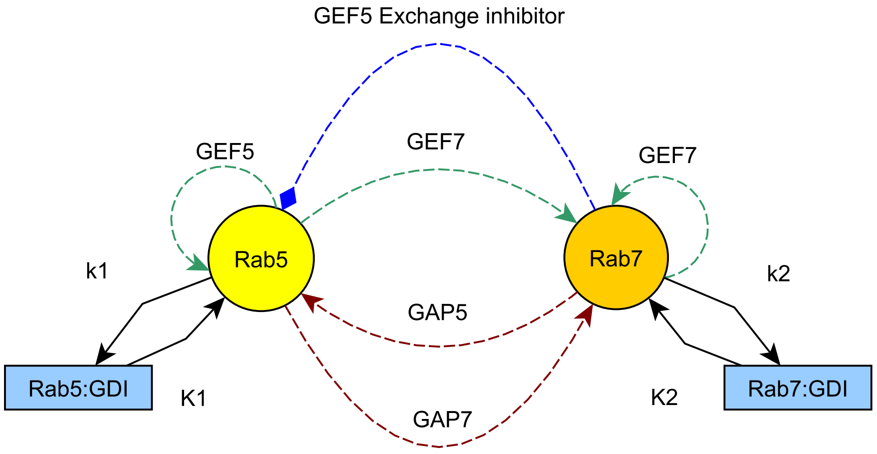


Fig. 2. A graphical representation of the Rab5-Rab7 interaction model structure as considered by del Conte-Zerial et al. [2]

Table 1. Part of the developed library of domain knowledge. Definition of the template entity Protein and the template root process.

```
template entity Protein {
  vars:
    GDP_bound_state_conc {range:<0,2>}, GTP_bound_state_conc {range:<0,2>},
    GEF, GAP, t;
  consts:
    GDI_dissociation_flux {range: <0.001, 4>},
    GDI_association_rate {range: <0.001, 4>};
}
template process Root(p1 : Protein, p2: Protein) {
  consts:
    td {range: <50,150>};
processes:
    GDI_GDP_membrane_interaction(p1), GDI_GDP_membrane_interaction(p2),
    GEFProcess(p1,p2), GEFCombined(p1,p2), GAPProcessPlus(p1,p2),
    GAPProcess(p2,p1);
equations:
    td(p1.GDP_bound_state_conc) = - p1.GEF*(p1.t/(p1.t+td))
                                  * p1.GDP_bound_state_conc
                                  + p1.GAP * p1.GTP_bound_state_conc,
    td(p1.GTP_bound_state_conc) =   p1.GEF * (p1.t/(p1.t+td))
                                  * p1.GDP_bound_state_conc
                                  - p1.GAP * p1.GTP_bound_state_conc,
    td(p2.GDP_bound_state_conc) = - p2.GEF * p2.GDP_bound_state_conc
                                  + p2.GAP * p2.GTP_bound_state_conc,
    td(p2.GTP_bound_state_conc) =   p2.GEF * p2.GDP_bound_state_conc
                                  - p2.GAP * p2.GTP_bound_state_conc;
}
// ...
```

Table 1 presents a fragment of the ProBMoT library for inductive process modeling of endocytosis. It contains a single template entity protein that corresponds to both of the protein domains of Rab5 and Rab7. The variables of the protein template correspond to the concentrations of the active-state and passive-state proteins. Furthermore, the root process template specifies the general form of the conversion model from Equation 1, where the interaction processes correspond to the interactions from Figure 2. Finally, the library contains specifications of all kinetic rate law alternatives considered for modeling individual reactions (interactions in Figure 2) that del Conte-Zerial et al. [2] considered in their work.

Note the presence of a time dependent term $\frac{t}{t+td}$ in the root process equations which is not present in Equation 1. In this term t corresponds to the model time and td is a parameter that needs to be fitted. Under the assumption that the model which describes the Rab5-Rab7 conversion switch is bistable, a change in a parameter is needed in order for a switch between the stable states of the model to happen. This term modifies the ke parameter of the GEF5 interaction in a similar fashion as reported in del Conte-Zerial et al. [2].

Table 2. Part of the developed library of domain knowledge. Definition of interaction processes with alternative forms.

```
template process GEFProcess(p1: Protein, p2: Protein){
  consts: ke{range:<0.001,4>},kf{range:<0.001,4>},kg{range:<0.001,4>},
    km{range:<0.001,4>},ki{range:<0.001,4>};
}
template process MMKinetics : GEFProcess {
  equations:
    p1.GEF = ke*p1.GTP_bound_state_conc/(kg + p1.GTP_bound_state_conc);
}
template process Sigmoidal_response : GEFProcess {
  equations:
    p1.GEF = ke/(1 + exp(kg - p1.GTP_bound_state_conc)*kf);
}
template process Exchange_inhibition : GEFProcess {
  equations:
    p1.GEF = ke*p1.GTP_bound_state_conc/(km*(1+p2.GTP_bound_state_conc/ki)
                                          + p1.GTP_bound_state_conc);
}
template process GEFCombined(p1: Protein, p2: Protein){
  consts: ke{range:<0.001,4>},kf{range:<0.001,4>},kg{range:<0.001,4>},
    km{range:<0.001,4>},ki{range:<0.001,4>},kE{range:<0.001,4>};
}
//..
template process GAPProcessPlus(p1: Protein, p2: Protein){
  consts: kh{range:<0.001,4>},kH{range:<0.001,4>},ky{range:<0.001,4>};
}
//..
template process GAPProcess : GAPProcessPlus{}
//..
```

Table 3. The incomplete model given to ProBMoT

```
incomplete model EndocytosisModel:EndocytosisLibrary;
entity rab5 : Protein {
  vars:
    GDP_bound_state_conc{role:endogenous; initial:null;},
    GTP_bound_state_conc{role:endogenous; initial:null;},
    GEF{role:endogenous}, GAP{role:endogenous}, t{role:exogenous};
  consts: GDI_dissociation_flux, GDI_association_rate;
}
entity rab7 : Protein {
  vars:
    GDP_bound_state_conc{role:endogenous; initial:null;},
    GTP_bound_state_conc{role: endogenous; initial: null;},
    GEF{role:endogenous}, GAP{role:endogenous},t{role:exogenous};
  consts: GDI_dissociation_flux, GDI_association_rate;
}
process root(rab5, rab7) : Root {
  processes:
    GDI_GDP_membrane_interaction5, GDI_GDP_membrane_interaction7,
    GEF5Process, GEF7Process, GAP5Process,GAP7Process;
}
process GDI_GDP_membrane_interaction5(rab5):GDI_GDP_membrane_interaction{
  processes: Dissociation_from_GDI5, Association_with_GDI5;
}
process GDI_GDP_membrane_interaction7(rab7):GDI_GDP_membrane_interaction{
  processes: Dissociation_from_GDI7, Association_with_GDI7;
}
process Dissociation_from_GDI5(rab5):Dissociation_from_GDI{}
process Association_with_GDI5(rab5):Association_with_GDI{}
process Dissociation_from_GDI7(rab7):Dissociation_from_GDI{}
process Association_with_GDI7(rab7):Association_with_GDI{}
process GEF5Process(rab5,rab7):GEFProcess{
  consts: ke,kf,kg,km,ki;
}
process GEF7Process(rab5,rab7):GEFCombined{
  consts: ke,kf,kg,km,ki;
}
process GAP5Process(rab5,rab7):GAPProcessPlus{
  consts: kh,kH,ky;
}
process GAP5Process(rab7,rab5):GAPProcess{
  consts: kh,kH,ky;
}
```

Table 2 presents an additional fragment of the library in which all possible alternative modeling choices for the GEF5 interaction (`GEFProcess` interaction) are described with their corresponding equation fragments. All other processes in the library are described in a similar hierarchical manner. Only the top level processes are presented in Table 2 while the alternatives are omitted.

The `GEFCombined` interaction process contains the possible alternatives for the GEF7 interaction. The `GAPProcess` interaction process contains the possible alternatives for the GAP7 interaction and the `GAPProcessPlus` interaction process contains the possible alternatives for the GAP5 interaction. The additional processes that are not listed in Table 2, namely the `GDI_GDP_membrane_interaction` contains the kinetic laws of the interaction between the protein domains and GDI which include association and dissociation, represented with black solid lines in Figure 2. Table 3 presents the incomplete (conceptual) model for the problem of modeling the Rab5-Rab7 conversion switch.

4 Modeling Endocytosis with ProBMoT

The process-based knowledge in the form of library of template entities and processes provides alternatives for modeling individual endocytosis processes. ProBMoT also takes as input an incomplete model which provides a general specification of the entities and processes involved in the Rab5-Rab7 conversion. Using the library ant the incomplete model from Table 3 ProBMoT enumerates all candidate model structures. The enumeration results in 126 candidate structures. Del Conte-Zerial et al. [2], consider only a subset of 54 structures and omit others. Note that ProBMoT automates the structure identification task and allows us to perform a complete experiment, where all 126 structures are considered.

Obtaining a complete model from the enumerated structure requires estimation of the parameter values. To this end, we use Differential Evolution as a parameter fitting method, which was configured according to the settings reported by Tashkova et al. [9]. The parameter ranges were set to the interval $[0, 2]$ for the initial values of the system variables ($r5$, $R5$, $r7$, and $R7$), to $[50, 150]$ for the td parameter in the root process, to $[10^{-3}, 4]$ for all the other model parameters, and to $[10^3, 10^5]$ for the output scaling parameter (see below).

Each model for each candidate parameter set in the process of parameter estimation is evaluated using the available data and the output of the model simulation. The parameter estimation is guided by an objective function based on root mean squared error (RMSE). The objective function for evaluating a model m is defined as in equation 2.

$$RMSEObjective(m) = \sum_{i=1}^{2} \sqrt{\frac{1}{N}\sum_{j=1}^{N}(x_i[j] - y_i[j])^2}, \qquad (2)$$

where $x_i[j]$ and $y_i[j]$ denote the simulated and measured (respectively) value of the observed variable i at time point j, and N denotes the number of the measurement time points in the dataset.

Depending on the candidate structure the total number of parameters that need to be estimated (initial values, model parameters and output scaling parameters) is different and varies between 17 and 25. The parameter fitting method was set to perform 20000 evaluations per unknown parameter. This resulted in 340000 to 500000 evaluations per model depending on the total number of unknown parameters.

The data provided for the experiment comprises of two time-series of measurements from del Conte-Zerial et al. [2]. The data was collected from three independent experiments (28 time courses), scaled and averaged. The data consist of 10,571 time points on the interval $[-5, 330]$ seconds. The measured time-series for Rab5 and Rab7 concentrations were manually aligned, so the conversion switch point is at time point 0.

Due to the limitation of the measurement equipment, only the total concentration of the Rab5 and Rab7 domain proteins can be observed. Therefore, the observed values at each time point correspond to $r_5 + R_5$ and $r_7 + R_7$. To deal with the limited observations and their scale, we define the model output as $K * (r_5 + R_5)$ and $K * (r_7 + R_7)$, where K denotes a scaling parameter. Additionally, we used the transformation $t \leftarrow t + 828.56$ on the time points from the data, shifting the conversion point to the real time so that our model simulations can be directly compared to the measured data [2,9].

4.1 Experimental Results and Discussion

The range of error for the obtained models is narrow. Many models have errors only slightly higher than the best model. The errors of all models have a mean equal to 3841.65 and a deviation equal to 1022.12. Additionally, the mean of the first 100 models is 3411 with a deviation of only 201.34.

Figure 3a provides a graphical representation of the best ranking model structure according to its *RMSEObjective* value after fitting to the data. It has an error of 3040.08.

Although the range of errors is narrow, the models differ by their structural complexity. We define the structural complexity of the model by the number of interactions (represented as arrows in Figures 2 and 3) present in the model.

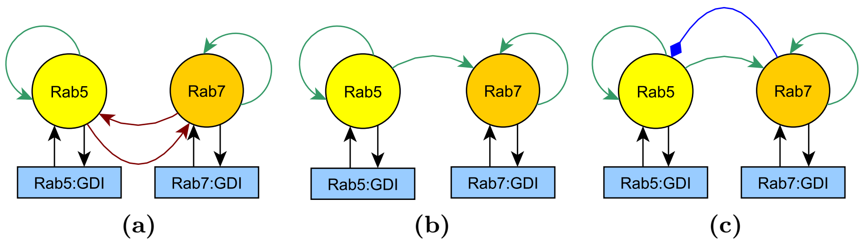


Fig. 3. (a) The best model structure according to the *RMSEObjective*. (b) The best model structure according to the BIC. (c) The model structure reported by del Conte-Zerial et al. [2].

Structurally, our best model (according to the *RMSEObjective* value) has similar complexity with the one reported by del Conte-Zerial et al. [2] (shown in Figure 3c). Given the narrow range of errors, we were interested in finding the best model with the lowest structural complexity that has a good fit to the data. In the literature [6], a function (Equation 3) based on the BIC (Bayesian Information Criterion) is commonly used to combine the complexity and the fit of the models.

$$BIC(m) = N \ln(MSE(m)) + k \ln(N) \tag{3}$$

In Equation 3, N represents the number of points in the data, k represents the complexity of the model and *MSE(m)* represents the mean squared error of the model simulation given the measured data.

Figure 3b presents the structure of the best ranked model using this function. This model structure has lower complexity than the best ranking model according to the *RMSEObjective* value, its error is 3145.05, and it was ranked as 13th by the *RMSEObjective* value. The cut-out switch structure reported by del Conte-Zerial et al. [2] (Figure 3c) has an error of 3735.98 and ranks 93rd according to the *RMSEObjective* value and 94th according to the BIC value. Structurally, the model ranked as best according to the BIC is more similar to the cut-out-switch structure than the model ranked as best according to the *RMSEObjective*. The only difference between the two is in the missing inhibitory feedback from Rab7 to Rab5.

Despite the observed error and rank difference, the best ranking models according to the used criteria and the model reported in [2], lead to similar simulated dynamics of the total density. The simulations fit well to the measured data as shown in Figure 4, where Model 3A denotes the best ranked model found by ProBMoT according to the *RMSEObjective* value, Model 3B denotes the best

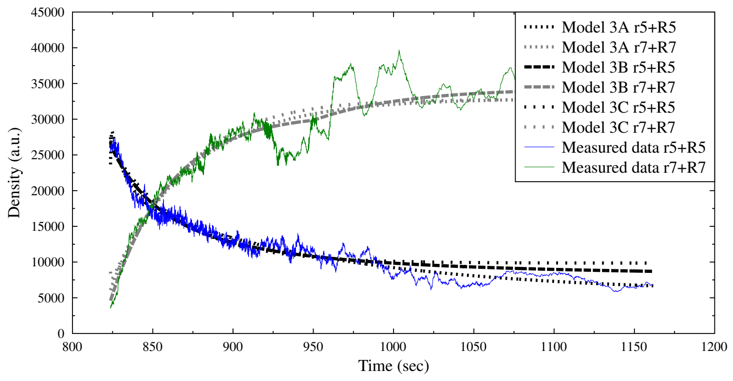
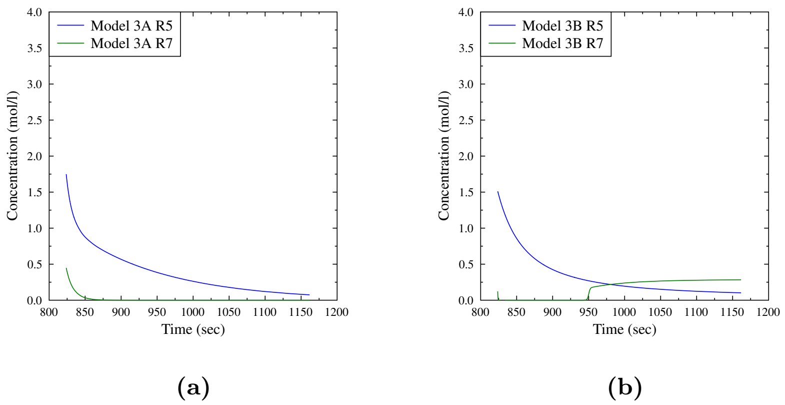


Fig. 4. Simulations of the total density of Rab5 and Rab7 in Model 3A, Model 3B and Model 3C, compared to the measured data

ranked model according to the BIC value, while Model 3C denotes the model reported in [2]. The structures of the models 3A, 3B and 3C correspond with the structures shown in Figures 3a, 3b and 3c.

Let us now examine in detail the simulated dynamics of the specific protein domains. The best model according to the *RMSEObjective* value does not properly model the expected switch in the $R5$ and $R7$ trajectories (see Figure 5). This is due to the missing triggering interaction between Rab5 and Rab7, which will activate Rab7. In the case of the best ranked model according to the BIC value, there is no mechanism of Rab5 removal catalysed by Rab7. The removal of Rab5 is only due to intrinsic hydrolysis. Although we observe switching behavior, the dynamics of the switch, and the time at which it happens, does not correspond to the ones observed in the data.



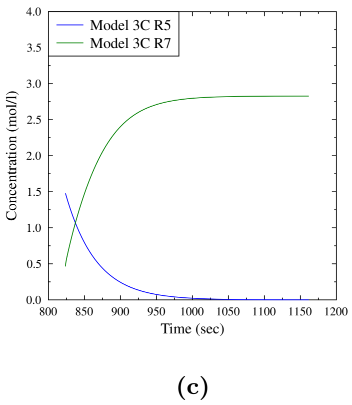


Fig. 5. Simulation of the dynamics of Rab-GTP concentrations of (a) Model 3A, (b) Model 3B and (c) Model 3C

Inspecting the model produced by ProBMoT, we noted that the fitted value of the *td* parameter in many of the models was on the lower or the upper bound of the fitting range. We thus performed another set of experiments using the same setup with only one change. We extended the fitting range of the parameter *td* in the time dependent term responsible for parameter perturbation from the initial range $[50, 150]$ to $[5, 195]$, thus increasing the space of possible parameter perturbations.

The range of errors for the obtained models remained narrow. The change in the setup resulted in a decrease of the error values of all models. The errors of all models in the new setup have a mean equal to 3327.06 and a deviation equal to 607.49. The mean of the first 100 models is 3113.56 with deviation equal to 148.

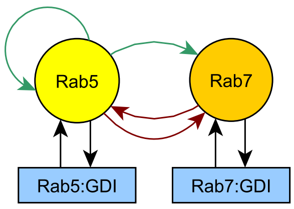


Fig. 6. The best ranked model structure according to both *RMSEObjective* and BIC values

The ranking of the models significantly changed and the new best model according to the *RMSEObjective* value is also the new best model according to the BIC value. Figure 6 presents the structure of the newly obtained best model (Model 6). The complete model expressed in the ProBMoT formalism is given in table 4. It contains the specific processes (reactions), their kinetics and the kinetic rate constants. It has an error of 2800.22. Structurally, although the model lacks the mechanism for sustaining the Rab7 concentration after Rab5 is removed from the system, the model is capable of producing switch like behaviour as it contains mechanisms for triggering activation of Rab7 and disposal of Rab5, as opposed to the previously obtained best models. It is also important to mention than according to the new rankings, the structure reported in the literature (Figure 3c) was fitted so its rank according to its *RMSEObjective* value (3072.01) improved to 46, while its rank according to its BIC value improved to 39.

As was the case in the previous experiments the best ranking model according to the used criteria and the model reported in the literature lead to similar simulated behaviour of the total density of the proteins. Again, the simulated behaviour fits well to the measured data (Figure 7). In Figure 7, Model 6 denotes the best ranking model, while Model 3C denotes the model reported by del Conte-Zerial et al. [2].

Table 4. The complete model 6 with fitted parameter values represented in the ProBMoT formalism

```
model EndocytosisModel : EndocytosisLibrary;
entity rab5:Protein{
  vars:
    GDP_bound_state_conc {role:endogenous; initial:1.9791404058513797;},
    GTP_bound_state_conc {role:endogenous; initial:1.3140551623312862;},
    GEF {role:endogenous;},GAP{role:endogenous;},t {role:exogenous;};
  consts:
    GDI_dissociation_flux = 0.05910840467192948,
    GDI_association_rate = 0.08397972050684883;
}
entity rab7:Protein{
  vars:
    GDP_bound_state_conc {role:endogenous; initial:0.20768919791767312;},
    GTP_bound_state_conc {role:endogenous; initial:0.15427162958067459;},
    GEF{role:endogenous;},GAP{role:endogenous;},t{role:exogenous;};
  consts:
    GDI_dissociation_flux = 0.1835340494246706,
    GDI_association_rate = 1.1655169518991926;
}
process root(rab5, rab7):Root{
  consts: td = 179.94021940272;
  processes:
    GDI_GDP_membrane_interaction5, GDI_GDP_membrane_interaction7,
    GEF5Process, GEF7Process, GAP5Process, GAP7Process;
}
process GDI_GDP_membrane_interaction5(rab5):GDI_GDP_membrane_interaction{
  processes: Dissociation_from_GDI5, Association_with_GDI5;
}
process GDI_GDP_membrane_interaction7(rab7):GDI_GDP_membrane_interaction{
  processes: Dissociation_from_GDI7, Association_with_GDI7;
}
process Dissociation_from_GDI5(rab5) : Dissociation_from_GDI {}
process Association_with_GDI5(rab5) : Association_with_GDI {}
process Dissociation_from_GDI7(rab7) : Dissociation_from_GDI {}
process Association_with_GDI7(rab7) : Association_with_GDI {}
process GEF5Process(rab5, rab7):MMKinetics{
  consts: ke=0.001, kg=2.3957792873725934,
}
process GEF7Process(rab5, rab7):NoAct_MM{
  consts:
    ke=0.06870234939183612,kg=0.11081576860190387,kE=3.5647410609632955;
}
process GAP5Process(rab5, rab7):MM{
  consts: kH=0.010407865344899033,ky=3.824554413479251;
  processes: intrinsic_Hydrolysis547;
}
process GAP7Process(rab7, rab5):MM{
  consts: kH=0.5475865295310145, ky=4.0;
  processes: intrinsic_Hydrolysis311;
}
process intrinsic_Hydrolysis311(rab7, rab5):Intrinsic_Hydrolysis{
  consts: kh=0.06910969838766903;
}
process intrinsic_Hydrolysis547(rab5, rab7):Intrinsic_Hydrolysis{
  consts: kh=0.003723987022345545;
}
// RMSEObjective=2800.22793923356 // K=8545.043326240191
```

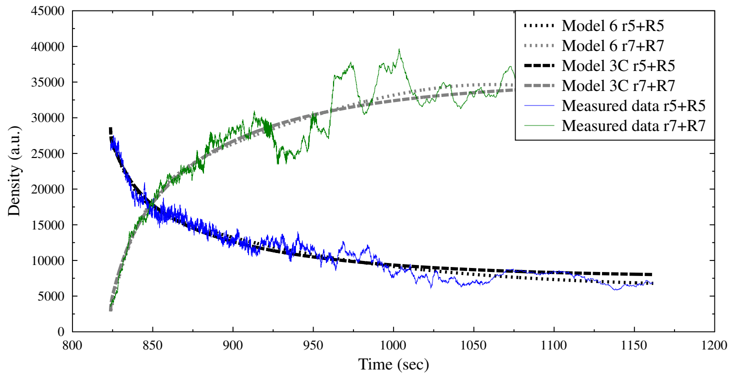


Fig. 7. Simulations of the total density of Rab5 and Rab7 in Model 6 and Model 3C, compared to the measured data

Looking at the dynamics of the Rab-GTP concentrations, we observe a switch-like behaviour in both the best model and the model reported in the literature (Figure 8).

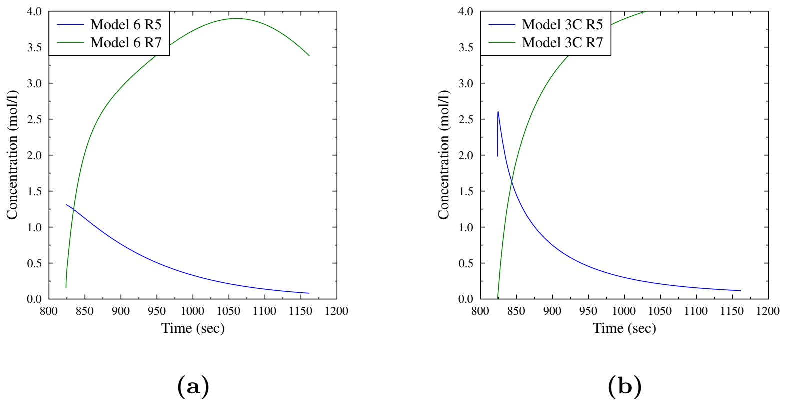


Fig. 8. Simulation of the dynamics of Rab-GTP concentrations of (a) Model 6 and (b) Model 3C

4.2 Conclusions

Using a function based on the sum of squared error as a single and only objective function guiding the parameter estimation may lead to a situation where discriminating between model structures can be a difficult task. The additional model selection criterion, based on the complexity of the models, can be used for better model discrimination given a narrow error range of the models.

In the case of modeling the Rab5-Rab7 switch in endocytosis using both of the considered criteria at first resulted in top ranked models which did not have a structure that can produce the desired switch-like behaviour. Due to the number of parameters that have to be fitted for one model, and the limited observability of the measured data the models can be overfitted to the data, and a meaningful model selection in this situation becomes a virtually impossible task.

In the second round of experiments, we allowed a larger range for only one parameter which is responsible for the parameter perturbation needed for achieving switching behaviour. The error range of the obtained models became even narrower, but both selection criteria ranked one model as the best. Structurally, the newly obtained model was able to reproduce switch like behaviour.

However, the nature of the objective function does not allow proper detection of complex dynamical behavior. Additional criteria which complement the information about the fit to the measured data gained by this objective function should be also considered. One such criterion is based on performing bifurcation and phase plane analysis [2]. Only after the model structure passes certain criteria related to these analyses, the comparison with the measured data is to be made for the final model selection.

5 Summary and Further Work

We encoded process-based knowledge for modeling endocytosis and used the IPM-based tool ProBMoT to model the endocytosis process. In particular, we focused on the process of Rab5-Rab7 protein domain conversion. We compared the obtained models with the results from manual modeling experiments [2].

We used different selection criteria in order to select the best model. The best obtained models with the automated approach differ from the model reported in the manual experiments. We found out that range of errors obtained for different model structures is narrow and we performed additional model selection based on criteria which take into account both the fit of the model to the data and its complexity. Due to the complex representation of the processes, the number of parameters that need to be fitted for each model and the limited observability of the measured data, some models which might not fulfil the structural requirements for the desired behavior can be overfitted to the data and be preferred by the model selection criteria. These findings give direction to further work in the process of automated modeling of endocytosis and the further development of the IPM approach.

Additional apriori filtering of the explored model structures is needed, based on the structural properties of the models, resulting in a reduced number of candidates to be fitted to the measured data. Note that also the objective function based on squared error, used for parameter estimation, might not be sufficient for discriminating among the model structures when modeling endocytosis. Going beyond the use of this kind of error measures as the only criterion for optimization is one possible direction for further work. The integration of information gained by performing bifurcation and phase plane analysis on the candidate models in the IPM approach, among the lines of del Conte-Zerial et al. [2], might improve our method and result in better discrimination among model structures.

Finally, the Rab5-Rab7 conversion, although important, is only one phase of the whole endocytosis process. The IPM approach can be also used to gain knowledge about the dynamics of other parts of the endocytic pathway. This knowledge can contribute to the development of a complete explanatory model of endocytosis.

Acknowledgements. We would like to gratefully acknowledge the financial support of the following institutions: The Slovene Human Resources Development and Scholarship Fund, the Slovenian Research Agency (Grant P2-0103), the European Commission (Grants ICT-2010-266722 and ICT-2011-287713), and the Operation no. OP13.1.1.2.02.0005 financed by the European Regional Development Fund (85%) and the Ministry of Education, Science, and Sport of Slovenia (15%).

References

1. Bridewell, W., Langley, P., Todorovski, L., Džeroski, S.: Inductive process modelling. Machine Learning 71, 109–130 (2008)
2. del Conte-Zerial, P., Brusch, L., Rink, J., Collinet, C., Kalaidzidis, Y., Zerial, M., Deutsch, A.: Membrane identity and GTPase cascades regulated by toggle and cut-out switches. Molecular Systems Biology 4 (2008)
3. Durillo, J.J., Nebro, A.J.: jmetal: A java framework for multi-objective optimization. Advances in Engineering Software 42, 760–771 (2011)
4. Džeroski, S., Todorovski, L.: Encoding and using domain knowledge on population dynamics for equation discovery. In: Magnani, L., Nersessian, N.J. (eds.) Logical and Computational Aspects of Model-Based Reasoning, pp. 227–247. Kluwer Academic Publishers (2002)
5. Džeroski, S., Todorovski, L.: Discovering dynamics. In: Proc. Tenth International Conference on Machine Learning, pp. 97–103. Morgan Kaufmann (1993)
6. Hastie, T., Tibshirani, R., Friedman, J.H.: The elements of statistical learning: data mining, inference, and prediction. Springer (2001)
7. Hindmarsh, A.C., Brown, P.N., Grant, K.E., Lee, S.L., Serban, R., Shumaker, D.E., Woodward, C.S.: SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. ACM Trans. Math. Softw. 31(3), 363–396 (2005)
8. Langley, P., Simon, H., Bradshaw, G., Zytkow, J.: Scientific Discovery: Computational Explorations of the Creative Processes. The MIT Press (1987)
9. Tashkova, K., Korošec, P., Šilc, J., Todorovski, L., Džeroski, S.: Parameter estimation with bio-inspired meta-heuristic optimization: modeling the dynamics of endocytosis. BMC Systems Biology 5(1), 159 (2011)
10. Čerepnalkoski, D., Taškova, K., Todorovski, L., Atanasova, N., Džeroski, S.: The influence of parameter fitting methods on model structure selection in automated modeling of aquatic ecosystems. Ecological Modelling 245, 136–165 (2012)

Fast Compression of Large-Scale Hypergraphs for Solving Combinatorial Problems

Takahisa Toda

ERATO MINATO Discrete Structure Manipulation System Project, Japan Science and Technology Agency, at Hokkaido University, Sapporo 060-0814, Japan
toda@erato.ist.hokudai.ac.jp, toda.takahisa@gmail.com

Abstract. We present a fast algorithm to compress hypergraphs into the data structure ZDDs. We furthermore analyze the computational complexity. Our algorithm uses multikey Quicksort given by Bentley and Sedgewick. By conducting experiments with various datasets, we show that our algorithm is significantly faster and requires much smaller memory than an existing method.

Keywords: hypergraph, binary decision diagram, data compression, sort, data mining, combinatorial problem.

1 Introduction

A *hypergraph* is a set family over a ground set, where each set is called a *hyperedge*. A hypergraph can represent a wide variety of information and thereby appears various areas in computer science. In data mining or knowledge discovery in databases, one of important tasks is to find all interesting patterns from databases. Mannila and Toivonen [1] presented a general framework for this problem. One of their results was to identify the class of problems in which patterns are representable as sets. In this class, patterns form a hypergraph.

A *zero-suppressed binary decision diagram* (ZDD) is a compressed data structure for hypergraphs. This was introduced by Minato [2] for solving combinatorial problems concerning hypergraphs. In many cases a ZDD has high compression efficiency, and furthermore many operations to manipulate hypergraphs such as intersection, union and difference are available. Since the time required for such set-theoretical operations is only related to ZDD sizes [3], a method to solve problems by using ZDDs is efficient especially when high compression efficiency is achieved. Although the use of ZDDs is not always a better approach, many results including [4–6] support that a ZDD-based method can be useful for some large-scale problems that are difficult to compute in a reasonable amount of time.

In the context above, Minato et al. [7] applied ZDDs to itemset mining problems by combining one of the most efficient frequent itemset generation algorithm, called LCM, and ZDDs. More specifically, this method LCM over ZDD constructs a ZDD in a bottom up fashion while receiving itemsets successively

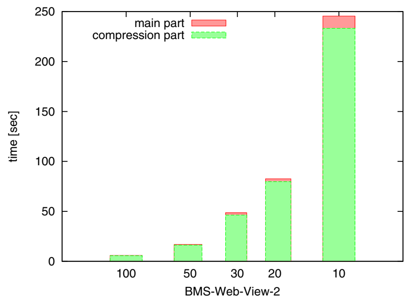


Fig. 1. The time comparison of the compression part and the remaining part in the transversal hypergraph computation. Each dataset consists of the complements of maximal frequent itemsets with support thresholds, generated from the real-life datasets "BMS-WebView2".

generated by LCM. Since a very large number of itemsets can be generated, storing and indexing them as a ZDD becomes useful in a data analysis phase to find some structural information underlying databases. Such analysis problems over ZDDs have been studied in [8–11].

ZDDs are not only useful in the post-process of LCM. Since there are many important computational problems concerning hypergraphs, ZDDs would be also useful for solving such problems. One of the most important problems is to compute the *transversal hypergraph* $\mathrm{Tr}(\mathcal{H})$ for a given hypergraph $\mathcal{H}$, where $\mathrm{Tr}(\mathcal{H})$ is a hypergraph which has the same ground set as $\mathcal{H}$ and which consists of all sets that can intersect every member of $\mathcal{H}$. This problem has many equivalent problems in various areas such as data mining, logic, artificial intelligence, etc. Hence, it has been extensively studied for years (see [12–16]). Recently, practically fast algorithms have been developed by many researchers. Among them, the algorithm given by Murakami and Uno [17] is known to be the fastest. In the previous study [6], we presented an algorithm to compute transversal hypergraphs with binary decision diagrams and experimentally showed that our ZDD-based algorithm is highly competitive with Murakami-Uno method. However, there is a drawback. As shown in Fig. 1, the compression of some datasets spent almost all time, although our algorithm was still significantly faster than the Murakami-Uno method. This implies that even if operations on ZDDs can be done quickly, the compression part can be a critical bottleneck. In order to overcome this situation, an efficient compression method is necessary.

In this paper, we present a fast algorithm to compress hypergraphs into ZDDs. A usual method is to construct a ZDD by repeating the union operation provided in a usual ZDD library (see for example [6, 18, 19]). Although it is easy to implement, unfortunately there are some drawbacks on performance. Yet another method was used in the LCM over ZDDs. As mentioned earlier, this method constructs a ZDD in a bottom up fashion while receiving sets successively generated by LCM. Since the focus of their work was a combination of LCM and ZDD, a compression procedure in a general setting was not treated and hence any performance on the bottom-up construction was not clarified. We would thus like to study a general construction problem. The basic idea of our algorithm is as follows. Like LCM over ZDDs, our algorithm constructs ZDDs in a bottom up fashion, and furthermore we accelerate it in cooperation with multikey Quicksort, which is a sort algorithm for strings given by Bentley and Sedgewick [20]. The performance of multikey Quicksort is well-analyzed. Thus we analyze only the computational complexity for the construction part. Furthermore, by conducting experimental comparison with many datasets, we show that our algorithm is significantly faster and requires much smaller memory than the usual method based on union operation.

This paper is organized as follows. In Section 2 we introduce the data structure ZDDs. In Section 3 we present an algorithm and analyze the computational complexity. Section 4 provides experimental results. We conclude in the final section.

2 A Compressed Data Structure for Hypergraphs

Since we use a special data structure for hypergraphs, we provide necessary notions and results in this section. On detailed description of this data structure, the reader is referred to [21, 22]. We identify hypergraphs with set families if the ground set is clear from the context.

2.1 Introduction to ZDDs

A *zero-suppressed binary decision diagram* (*ZDD*) is a graph-representation for hypergraphs. Figure 2 shows an example of ZDD. The node at the top is called the *root*. Each internal node has the three fields V, LO, HI. The field V holds an element in a ground set, where for simplicity we suppose that a ground set consists of positive numbers. The fields LO and HI point to other nodes, which are called LO and HI *children*, respectively. The arc to a LO child is called a LO *arc* and illustrated by a dashed arrow, while the arc to a HI child is called a HI *arc* and illustrated by a solid arrow. There are only two terminal nodes $\top$ and $\bot$.

For efficient compression, ZDDs satisfy the following two conditions. They must be *ordered*: if a node u points to an internal node v, then $\mathrm{V}(u) < \mathrm{V}(v)$. They must be *reduced*: the following two reduction operations can not be applied.

1. For each internal node u whose HI arc points to $\bot$, redirect all the incoming arcs of u to the LO child, and then eliminate u (Fig. 3(a)).
2. For any nodes u and v, if the subgraphs rooted by u and v are equivalent, then share the two subgraphs (Fig. 3(b)).

We can understand ZDDs as follows. Given a ZDD, each path from the root to $\top$ corresponds to a set in such a way that an element k is included in the set if a path contains the HI arc of a node with label k; otherwise, k is excluded.

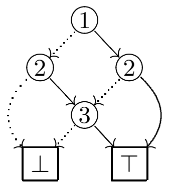
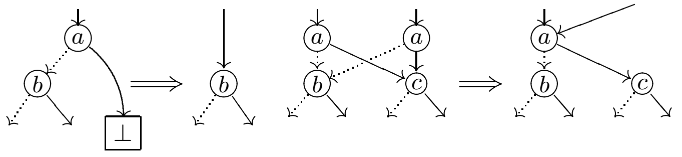


Fig. 2. The ZDD for the set family $\{\{2,3\},\{1,3\},\{1,2\}\}$



(a) Node elimination

(b) Node sharing

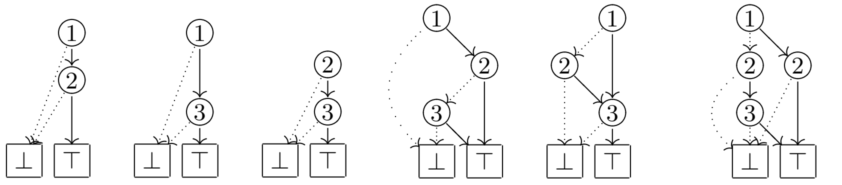
Fig. 3. Reduction rules on ZDDs

For example, in Fig. 2, the paths ① ⇢ ② → ③ → ⊤, ① → ② ⇢ ③ → ⊤, and ① → ② → ⊤ correspond to $\{2,3\}$, $\{1,3\}$ and $\{1,2\}$, respectively. Note that although the node ③ does not appear in the last path, the node elimination rule implies that the HI arc of ③ points to $\bot$, and thus 3 is excluded.

It is known (see for example [2, 21]) that for any ground set V, every hypergraph on V corresponds to a unique ZDD if the order of elements in V is fixed. ZDD nodes are maintained by a hash table, called a *uniquetable*, so that for a triple (k,l,h) of a node label and two ZDD nodes, there is a unique ZDD node p with $\mathrm{V}(p) = k$, $\mathrm{LO}(p) = l$, and $\mathrm{HI}(p) = h$. Given a triple (k,l,h), the function zdd_unique returns an associated node in the uniquetable if exists; otherwise, create a new node p such that $\mathrm{V}(p) = k$, $\mathrm{LO}(p) = l$, and $\mathrm{HI}(p) = h$; register p to the uniquetable and return p. A uniquetable guarantees that two nodes are different if and only if the subgraphs rooted by them represent different set families. Thus, for example, equivalence checking of set families can be done in constant time.

2.2 An Existing Construction Method

An existing method to construct ZDDs is described below. Let V be a ground set and let $\mathcal{E} := \{U_1,\dots,U_m\}$ be a set family on V. We denote by $Z(\mathcal{S})$ the ZDD for a set family $\mathcal{S}$. For each set U_i, we construct $Z(\{U_i\})$. We then obtain $Z(\{U_1,\dots,U_i\})$ by applying union operation to $Z(\{U_1,\dots,U_{i-1}\})$ and $Z(\{U_i\})$ (see Fig. 4). As argued in [6], this construction method requires $O(|\mathcal{E}|\cdot|V|)$ time.



(a) A (b) B (c) C (d) $A \cup B$ (e) $B \cup C$ (f) $C \cup A$

Fig. 4. Examples of union operation, where $A = \{\{1,2\}\}$, $B = \{\{1,3\}\}$, $C = \{\{2,3\}\}$. The ZDD for $A \cup B \cup C$ is given in Fig. 2.

Since union operation is provided in a usual ZDD library, this method is easy to implement, however there are some drawbacks.

– A practical efficiency depends on the order in which sets are added.
– The worst-case time depends on the size of a ground set even if the size of a set tends to be much smaller than the size of a ground set.
– Many ZDD nodes created during the computation may not appear in an output ZDD, which can cause an explosion of the size of uniquetable.

We remark that a method for constructing minimal, deterministic, acyclic finite-state automata from sorted data was studied in [23]. Although this method treats a different data structure from ZDDs, it is similar to the method based on union operation described above except that strings are added in lexicographic order.

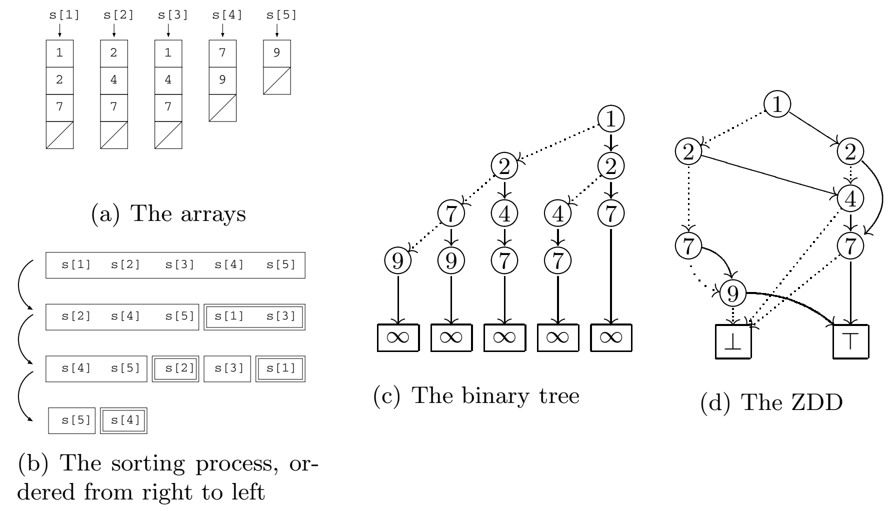
3 Algorithm

In this section, we first observe that ZDDs are isomorphic to processes of sorting sets and then present our algorithm.

Figure 5 shows an example of such a correspondence. Suppose that we are given hypergraph $(V, \mathcal{E})$, where V consists of positive numbers. Each hyperedge $U \in \mathcal{E}$ is represented as the array of numbers in U, ordered in increasing order. The d-th value of U always means the d-th smallest number in U. For the sake of simplicity, we assume that every hyperedge contains $+\infty$. We sort all hyperedges in lexicographic order by recursively partitioning $\mathcal{E}$ into the equal part $\mathcal{E}_=$ and the greater part $\mathcal{E}_>$ for the minimum value v at position d of hyperedges. Note that the following procedure should be initially started with $d = 1$.

```
function SORT($\mathcal{E}, d$)
    if $|\mathcal{E}| \leq 1$ then
        return;
    end if
    $v \leftarrow$ the minimum value among the $d$-th values of sets in $\mathcal{E}$;
    if $v = +\infty$ then
        return;
    end if
    $\mathcal{E}_= \leftarrow$ the set of hyperedges with the $d$-th values equal to $v$;
    $\mathcal{E}_> \leftarrow \mathcal{E} \setminus \mathcal{E}_=$;
    sort($\mathcal{E}_=, d+1$); sort($\mathcal{E}_>, d$);
end function
```

The sorting process can then be represented as an ordered binary tree such that each node holds a partitioning value v; the left and the right children of a node correspond to sort($\mathcal{E}_>, d$) and sort($\mathcal{E}_=, d+1$), respectively. When we consider the left and the right children of a node as the LO and the HI children, this binary tree satisfies the order condition for ZDD nodes, and furthermore it has an irreducible form with respect to the node elimination rule, because $v < +\infty$ implies $\mathcal{E}_= \neq \emptyset$. By sharing equivalent subgraphs, we obtain the corresponding



(a) The arrays

(b) The sorting process, ordered from right to left

(c) The binary tree

(d) The ZDD

Fig. 5. The process of sorting $s[1], \ldots, s[5]$ and the corresponding ZDD

ZDD. Note that this reduction does not make distinct trees the same ZDD. Therefore if two binary trees represent different sorting processes, then they must correspond to different hypergraphs, thus different ZDDs. Furthermore, every ZDD on V corresponds to a hypergraph on V. In this sense, if a ground set V is fixed, there is a bijective correspondence between ZDDs on V and binary trees representing sorting processes of hypergraphs on V.

This sort algorithm has a close relation to the ternary partitioning algorithm, called *multikey Quicksort*, given by Bentley and Sedgewick [20]. Their algorithm sorts multikey data in a similar way, but there are two different points. Firstly, a partitioning value v need not be minimum and there are many ways, from computing the true median to choosing a random value. Secondly, it partitions $\mathcal{E}$ into the three parts $\mathcal{E}_<$, $\mathcal{E}_=$ and $\mathcal{E}_>$, where $\mathcal{E}_<$, $\mathcal{E}_=$ and $\mathcal{E}_>$ consists of hyperedges with the d-th values smaller than, equal to and greater than v, respectively. An efficient partitioning method is important. The performance of multikey Quicksort is well-analyzed. For the sake of convenience, we extract the following two theorems from [20], where c denotes a constant. A worst-case and an expected-time algorithms that establish $c = 3$ and $c = 3/2$ are given in [24] and [25], respectively.

Theorem 1. *If multikey Quicksort partitions around a median computed in* cn *comparisons, it sorts* n k*-vectors in at most* $cn(\log n + k)$ *scalar comparisons.*

Let H_n denote the harmonic numbers, given as $H_n = \sum_{1 \le i \le n} 1/i$.

Theorem 2. *A multikey Quicksort that partitions around the median of* $2t + 1$ *randomly selected elements sorts* n k*-vectors in at most* $2nH_n/(H_{2t+2} - H_{t+1}) + O(kn)$ *expected scalar comparisons.*

When we sort sets by using multikey Quicksort, the number of comparisons does not depend on the size of a ground set. Thus it is effective especially when a set family is sparse. Since multikey Quicksort is based on a ternary partitioning method, unfortunately it is isomorphic to ternary search trees and does not generate a ZDD directly. For this, our approach consists of the following two parts.

1. We sort sets by using multikey Quicksort.
2. We then construct a ZDD, based on the binary partitioning method above.

Algorithm 1 shows the construction part.

Algorithm 1. Compute the ZDD for a set family $\mathcal{E}$

```
function ZCOMP($\mathcal{E}, d$)
    if $\mathcal{E} = \emptyset$ then
        return $\bot$;
    end if
    $v \leftarrow$ the minimum value among the $d$-th values of sets in $\mathcal{E}$;
    if $v = +\infty$ then
        return $\top$;
    end if
    $\mathcal{E}_= \leftarrow$ the set of hyperedges with the $d$-th values equal to $v$;
    $\mathcal{E}_> \leftarrow \mathcal{E} \setminus \mathcal{E}_=$;
    $hi \leftarrow$ zcomp($\mathcal{E}_=, d + 1$); $lo \leftarrow$ zcomp($\mathcal{E}_>, d$);
    return zdd_unique($v, lo, hi$);
end function
```

Theorem 3. *Suppose that an input set family $\mathcal{E}$ is given as an array of hyperedges sorted in lexicographic order. Algorithm 1 can be implemented to run in time proportional to $|\mathcal{E}| \log(\sum_{U \in \mathcal{E}} |U|/|\mathcal{E}|) + N$, where N denotes the number of recursive calls (equivalently, the number of nodes in the binary partitioning process). The required space is proportional to the size of an output ZDD.*

Proof. Since $\mathcal{E}$ is already sorted, in order to partition $\mathcal{E}$ to $\mathcal{E}_=$ and $\mathcal{E}_>$, it is sufficient to find the last hyperedge whose d-th values equal v. This can be efficiently done by skipping as many hyperedges as possible. To do this, consider the following procedure.

1. Let $i := 0$.
2. If the last hyperedge has v at position d, then return it.
3. While $2^i \leq |\mathcal{E}|$ and the d-th value equals v, search the 2^i-th hyperedge and then increment i by one.
4. Let $j := |\mathcal{E}|$ if the first condition breaks; otherwise, let $j := 2^i$.
5. Find a desired hyperedge U in the range from the 1st to the j-th hyperedges by using binary search; return U.

If all hyperedges have v at position d, then by the 2nd step the procedure above requires only constant time. Otherwise, since the 3rd and the 5th steps are over in $O(\lceil \log |\mathcal{E}_=| \rceil)$ steps, the whole procedure requires $O(\log |\mathcal{E}_=|)$ time. Let M be the total time for the latter case over the whole computation. Since the d-th values equal to v are not examined in later recursive calls, it follows that the sum of all sizes $|\mathcal{E}_=|$ in the latter case is at most the sum of all hyperedge sizes $\sum_{U \in \mathcal{E}} |U|$. Furthermore, the number of times the latter case occurs is $|\mathcal{E}| - 1$. Indeed, since binary partitioning process corresponds to a binary tree and the latter case in the above procedure corresponds to a node with two children[1], the number of times the latter case occurs is at most the number of leaves minus 1. From Jensen's inequality, we can easily derive $M \leq (|\mathcal{E}| - 1) \log(\sum_{U \in \mathcal{E}} |U|/(|\mathcal{E}| - 1))$, where without loss of generality we can assume $|\mathcal{E}| > 1$. On the other hand, the number of times the former case occurs is at most the number of recursive calls N. Therefore, we conclude that our algorithm requires time proportional to $|\mathcal{E}| \log(\sum_{U \in \mathcal{E}} |U|/|\mathcal{E}|) + N$.

Since Algorithm 1 constructs an output ZDD in a bottom up fashion, all nodes requested by zdd_unique appear in the output ZDD. Note that since the maximum depth of a recursive function call corresponds to the maximum length of a path in an output ZDD, it is clear that the space required to keep track of the recursive function calls is dominated by an output ZDD size. $\square$

In the construction algorithm given above, since we access only entries that are necessary to construct an output ZDD, in many cases our algorithm would be efficient. However there is a case in which the number of recursive calls N equals the sum of all hyperedge sizes $\sum_{U \in \mathcal{E}} |U|$, as illustrated in Fig 6. For such an input, our algorithm requires as the same time as for accessing all entries of input hyperedges. Thus, since $|\mathcal{E}| \log(\sum_{U \in \mathcal{E}} |U|/|\mathcal{E}|) \leq \sum_{U \in \mathcal{E}} |U|$, the worst-case time can be also given as $\sum_{U \in \mathcal{E}} |U|$. This has a simpler form than the one that we gave in the theorem. However, it only expresses the time for quite nasty hypergraphs, which seems rare to be given. Since we want to emphasize that not all entries need to be accessed if N is not so large, we used N as a parameter in the theorem.

As variants of the ZDD construction method above, changing sort algorithm or executing the sort part and the construction part simultaneously would be interesting.

4 Experiment

Implementation and Environment. We implemented our algorithm and the naive method based on union operation (presented in Section 2.2) in C. Although we implemented the competitor (the naive method) by ourselves, the algorithm is so simple that there seem almost no differences between implementations. In our programs, we used the BDD Package Sapporo-Edition-1.0 developed by Minato, in which ZDDs are available and various basic operations for ZDDs

[1] The former case corresponds in turn to a node with only one child.

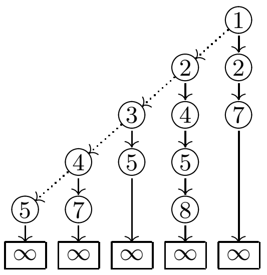


Fig. 6. A binary tree which has drooping linear lists

are provided. Our code is released in [26]. The (non-tuned) implementation of multikey Quicksort was obtained from [27]. All experiments were performed on a 2.67GHz Xeon®E7-8837 with 1.5TB RAM, running SUSE Linux Enterprise Server 11. We compiled our code with version 4.3.4 of the gcc compiler.

Problem Instances. We used many kinds of instances from different areas. We included very large instances, because high-performance machines with hundreds of GB of RAM are recently available and large-scale datasets have increasingly come into use [28].

The following two types of instances were obtained from the Hypergraph Dualization Repository [29], where n denotes an instance parameter. These instances have been commonly used in the experimental evaluation of hypergraph transversal computation (see for example [17]).

- BMS-WebView-2 (bms(n)): a hypergraph such that the size of a ground set is 3,341 and each hyperedge is the complement of a set of maximal frequent itemsets with support threshold n, of real-life datasets "BMS-WebView2" taken from the Frequent Itemset Mining Dataset Repository.
- Uniform random (rand(n)): a hypergraph such that the size of a ground set is 50 and the number of hyperedges is 1,024,000 and each vertex is included in a hyperedge in the probability $n/10$, generated by Prof. Alain Bretto.

Many other instances are available in that repository, however we do not present their experimental results because it took little time to compress.

The following three types of instances were obtained from the Frequent Itemset Mining Dataset Repository. These instances have been extensively used in data mining community.

- T40I10D100K: this data set was generated using the generator from the IBM Almaden Quest research group.
- retail: this dataset contains the (anonymized) retail market basket data from an anonymous Belgian retail store, donated by Tom Brijs.
- accidents: this dataset contains (anonymized) traffic accident data, donated by Karolien Geurts.

Table 1. Comparison of running time and maximum memory usage

	time (sec)			memory (G byte)		
	naive	sort+naive	zcomp	naive	sort+naive	zcomp
T40I10D100K	7.32	7.53	2.53	2.00	2.07	0.63
retail	15.59	18.85	0.34	4.18	4.42	0.16
accidents	19.36	18.43	3.84	4.19	4.39	1.19

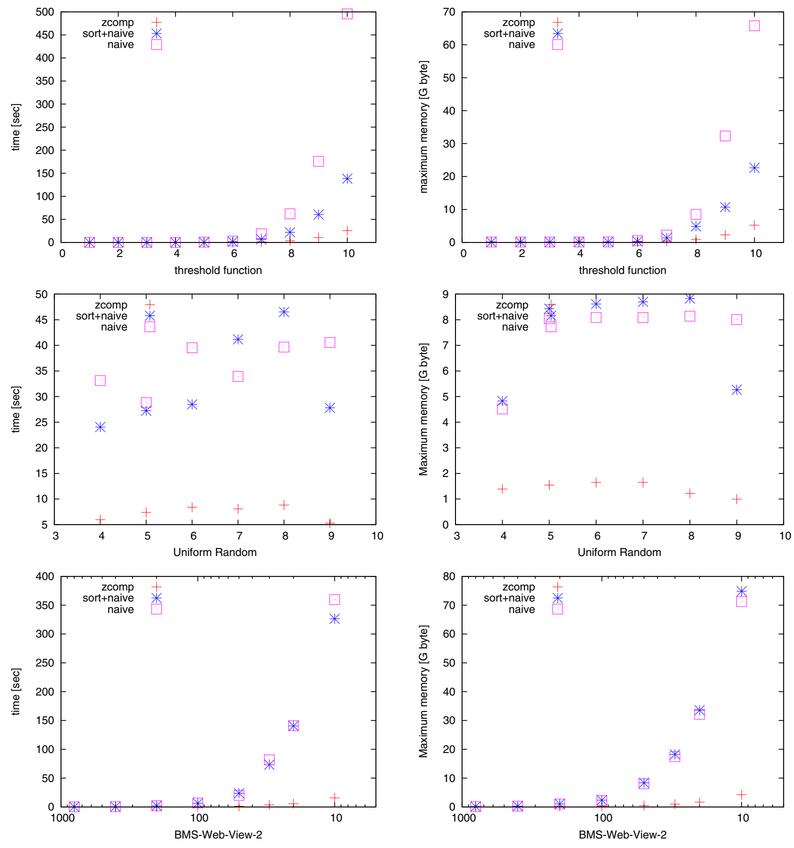


Fig. 7. Comparison of running time and maximum memory usage, where the horizontal coordinate of a point represents an instance parameter n

We furthermore generated instances of the following type.

– Threshold function (TH(n)): a hypergraph whose hyperedges correspond to prime conjunctive normal forms of the threshold function with 30 variables that returns 1 if at least n variables are 1. The number of hyperedges is $\binom{30}{n-1}$.

Threshold functions are known as a familiar example of a monotone Boolean function. In particular, if the parameter n is half of the number of variables, then they are known to be majority functions. We made these instances because we need very large-scale hypergraphs which takes much time even to read.

Data Format. All instances are given as data files with the following format: each row corresponds to a hyperedge, and entries are non-zero positive numbers (and less than or equal to the maximum number allowed in a BDD package); the entries in a row are sorted in increasing order and separated by a white space. Some data sets obtained from the Frequent Itemset Mining Dataset Repository were formated, since they did not have a correct form.

Comparison of Algorithms. We compared our algorithm (zcomp), the naive method based on union operation (naive), and the combination of multikey Quicksort and the naive method (sort+naive). Note that zcomp means the combination of multikey Quicksort and Algorithm 1. The results are shown in Fig. 7 and Table 1. All these results contain the cost required to read input datasets. In this experiment, the rows in data files were rearranged at random. The experiment shows that our algorithm is significantly faster and requires much smaller memory than the naive method.

5 Conclusion and Future Work

We have presented a new algorithm for constructing ZDDs for hypergraphs. The basic idea is to sort hyperedges by using multikey Quicksort and then construct a ZDD in a bottom up fashion. Since the performance of multikey Quicksort has been well-analyzed, we have analyzed only the computational complexity of the construction part. We have conducted experiments with various datasets, and the experiments have shown that our algorithm is significantly faster and requires much smaller memory than an existing method based on union operation. As demonstrated in previous studies, a ZDD-based framework can be considered as one of the most efficient approach to handle large-scale hypergraphs. Our algorithm would accelerate computations based on this framework.

A future work is to compare our compression with other techniques based on prefix-tree structure. In order to accelerate our algorithm further, it is also interesting to parallelize our algorithm by parallel sorting. For this, it is necessary to develop a parallel ZDD library. This paper only deals with families without duplicate sets. Since transaction databases generally contain duplicate transactions, it is also interesting to extend our algorithm so that it can compress families of sets with repetition into ZDD vectors [18] or algebraic decision diagrams [30].

Acknowledgments. The author would like to thank Professor Shin-ichi Minato, Professor Hiroki Arimura, Mr. Shuhei Denzumi, and the anonymous reviewers for their helpful suggestions and comments.

References

1. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery 1, 241–258 (1997)
2. Minato, S.: Zero-suppressed BDDs for set manipulation in combinatorial problems. In: Proceedings of 30th ACM/IEEE Design Automation Conference (DAC 1993), pp. 272–277 (June 1993)
3. Yoshinaka, R., Kawahara, J., Denzumi, S., Arimura, H., Minato, S.: Counterexample to the long-standing conjecture on the complexity of BDD binary operations. Information Processing Letters 112, 636–640 (2012)
4. Coudert, O.: Solving graph optimization problems with ZBDDs. In: Proceedings of the 1997 European Conference on Design and Test (EDTC 1997), pp. 224–228 (March 1997)
5. Kiyomi, M., Okamoto, Y., Saitoh, T.: Efficient enumeration of the directed binary perfect phylogenies from incomplete data. In: Klasing, R. (ed.) SEA 2012. LNCS, vol. 7276, pp. 248–259. Springer, Heidelberg (2012)
6. Toda, T.: Hypergraph transversal computation with binary decision diagrams. In: Bonifaci, V., Demetrescu, C., Marchetti-Spaccamela, A. (eds.) SEA 2013. LNCS, vol. 7933, pp. 91–102. Springer, Heidelberg (2013)
7. Minato, S.I., Uno, T., Arimura, H.: LCM over ZBDDs: Fast generation of very large-scale frequent itemsets using a compact graph-based representation. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 234–246. Springer, Heidelberg (2008)
8. Minato, S.: Finding simple disjoint decompositions in frequent itemset data using Zero-suppressed BDDs. In: Proceedings of IEEE ICDM Workshop on Computational Intelligence in Data Mining, pp. 3–11 (2005)
9. Minato, S.: A fast algorithm for cofactor implication checking and its application for knowledge discovery. In: Proceedings of IEEE 8th International Conference on Computer and Information Technology (CIT 2008), pp. 53–58 (2008)
10. Minato, S.-I.: Symmetric Item Set Mining Based on Zero-Suppressed BDDs. In: Todorovski, L., Lavrač, N., Jantke, K.P. (eds.) DS 2006. LNCS (LNAI), vol. 4265, pp. 321–326. Springer, Heidelberg (2006)
11. Toda, T.: Extracting co-occurrence relations from ZDDs. Algorithms 5(4), 654–667 (2012)
12. Eiter, T., Makino, K.: Abduction and the dualization problem. In: Grieser, G., Tanaka, Y., Yamamoto, A. (eds.) DS 2003. LNCS (LNAI), vol. 2843, pp. 1–20. Springer, Heidelberg (2003)
13. Bailey, J., Stuckey, P.J.: Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In: Hermenegildo, M., Cabeza, D. (eds.) PADL 2005. LNCS, vol. 3350, pp. 174–186. Springer, Heidelberg (2005)
14. Eiter, T., Gottlob, G.: Hypergraph transversal computation and related problems in logic and AI. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 549–564. Springer, Heidelberg (2002)
15. Eiter, T., Makino, K., Gottlob, G.: Computational aspects of monotone dualization: A brief survey. Discrete Applied Mathematics 156, 2035–2049 (2008)

16. Nourine, L., Petit, J.M.: Extending set-based dualization: Application to pattern mining. In: Proceedings of 20th European Conference on Artificial Intelligence (ECAI 2012), pp. 630–635 (August 2012)
17. Murakami, K., Uno, T.: Efficient algorithms for dualizing large-scale hypergraphs. In: Proceedings of the Meeting on Algorithm Engineering & Experiments (ALENEX 2013), pp. 1–13 (January 2013)
18. Minato, S., Arimura, H.: Efficient method of combinatorial item set analysis based on Zero-Suppressed BDDs. In: Proceedings of IEEE/IEICE/IPSJ International Workshop on Challenges in Web Information Retrieval and Integration, pp. 3–10 (April 2005)
19. Salleb, A., Vrain, C.: Estimation of the density of datasets with decision diagrams. In: Hacid, M.-S., Murray, N.V., Raś, Z.W., Tsumoto, S. (eds.) ISMIS 2005. LNCS (LNAI), vol. 3488, pp. 688–697. Springer, Heidelberg (2005)
20. Bentley, J., Sedgewick, R.: Fast algorithms for sorting and searching strings. In: Proceedings of 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1997), pp. 360–369 (January 1997)
21. Knuth, D.: The Art of Computer Programming, vol. 4a. Addison-Wesley Professional, New Jersey (2011)
22. Minato, S.: Techniques of BDD/ZDD: Brief history and recent activity. IEICE Transactions on Information and Systems E96-D(7), 1419–1429 (2013)
23. Daciuk, J., Watson, B.W., Mihov, S., Watson, R.E.: Incremental construction of minimal acyclic finite-state automata. Computational Linguistics 26(1), 3–16 (2000)
24. Schoenhage, A., Paterson, M., Pippenger, N.: Finding the median. Journal of Computer and Systems Sciences 13, 184–199 (1976)
25. Floyd, R., Rivest, R.: Expected time bounds for selection. Communications of the ACM 18(3), 165–172 (1975)
26. Toda, T.: ZCOMP: Fast Compression of Hypergraphs into ZDDs, `http://kuma-san.net/code/zcomp.html` (accessed on July 11, 2013)
27. Bentley, J., Sedgewick, R.: Fast algorithms for sorting and searching strings, `http://www.cs.princeton.edu/~rs/strings/` (accessed on April 7, 2013)
28. Buehrer, G., Parthasarathy, S., Tatikonda, S., Kurc, T., Saltz, J.: Toward terabyte pattern mining: an architecture-conscious solution. In: Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP 2007), pp. 2–12. ACM, New York (2007)
29. Murakami, K., Uno, T.: Hypergraph dualization repository, `http://research.nii.ac.jp/~uno/dualization.html` (accessed on January 19, 2013)
30. Bahar, R., Frohm, E., Gaona, C., Hachtel, G., Macii, E., Pardo, A., Somenzi, F.: Algebric decision diagrams and their applications. Formal Methods in System Design 10(2-3), 171–206 (1997)

Semantic Data Mining of Financial News Articles

Anže Vavpetič[1,2], Petra Kralj Novak[1], Miha Grčar[1], Igor Mozetič[1], and Nada Lavrač[1,2,3]

[1] Jožef Stefan Institute, Ljubljana, Slovenia
[2] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
[3] University of Nova Gorica, Nova Gorica, Slovenia
anze.vavpetic@ijs.si

Abstract. Subgroup discovery aims at constructing symbolic rules that describe statistically interesting subsets of instances with a chosen property of interest. Semantic subgroup discovery extends standard subgroup discovery approaches by exploiting ontological concepts in rule construction. Compared to previously developed semantic data mining systems SDM-SEGS and SDM-Aleph, this paper presents a general purpose semantic subgroup discovery system Hedwig that takes as input the training examples encoded in RDF, and constructs relational rules by effective top-down search of ontologies, also encoded as RDF triples. The effectiveness of the system is demonstrated through an application in a financial domain with the goal to analyze financial news in search for interesting vocabulary patterns that reflect credit default swap (CDS) trend reversal for financially troubled countries. The approach is showcased by analyzing over 8 million news articles collected in the period of eighteen months. The paper exemplifies the results by showing rules reflecting interesting news topics characterizing Portugal CDS trend reversal in terms of conjunctions of terms describing concepts at different levels of the concept hierarchy.

Keywords: semantic data mining, subgroup discovery, ontology, credit default swap, financial crisis.

1 Introduction

This paper addresses the task of subgroup discovery, first defined by Klösgen [1] and Wrobel [2]. The goal of SD is to find subgroups of instances that are statistically interesting according to some property of interest for a given population of instances. SD is commonly described as being in the intersection of predictive and descriptive data mining as it is used for descriptive rule learning although the rules are induced from class-labeled data. Patterns discovered by subgroup discovery methods (called subgroup descriptions) are rules of the form `Class` $\leftarrow$ `Conditions`, where the condition part of the rule is a logical conjunction of features (items, attribute values) or a conjunction of logical literals that are characteristic for a selected class of instances.

It is well known from the literature on inductive logic programming (ILP) [3, 4] and relational data mining (RDM) [5] that the performance of data mining methods can be significantly improved if additional relations among the data objects are taken into account. In other words, the knowledge discovery process can significantly benefit from the domain (background) knowledge.

A special form of background knowledge, which has not been exploited in the original ILP and RDM literature, are ontologies. Ontologies are consensually developed domain models that formally define the semantic descriptors and can act as means of providing additional information to machine learning (data mining) algorithms by attaching semantic descriptors to the data. Such domain knowledge is usually represented in a standard format which encourages knowledge reuse. Two popular formats are the Web Ontology Language (OWL) for ontologies and the Resource Description Framework (RDF) triplets for other structured data. The RDF data model is simple, yet powerful. A representation of the form *subject-predicate-object* ensures the flexibility of the data structures, and enables the integration of heterogeneous data sources. Data can be directly represented in RDF or (semi-)automatically translated from propositional representations to RDF as graph data. Consequently, more and more data from public relational databases are now being translated into RDF as linked data. In this way, data items from various databases can be easily linked and queried over multiple data repositories through the use of semantic descriptors provided by the supporting ontologies encoding the domain models and knowledge.

The process of exploiting formal ontologies within the process of data mining, called Semantic Data Mining (SDM), was formalized by Vavpetič and Lavrač [6]. Early work in using ontologies in machine learning and data mining is due to Kietz [7] who extended the standard learning bias used in ILP with description logic (DL) in his CLARIN-DL system. More recently, Lehmann and Haase [8] defined a refinement operator in a variant of DL, but considered only the construction of consistent and complete hypotheses. Lawrynowicz and Potoniec [9] introduced an algorithm for frequent concept mining in another variant of DL. Combining web mining and the semantic web was proposed by Berendt et al. [10]. Early work on this topic is due to Lisi et al. [11, 12], proposing an approach to mining the semantic web by using a hybrid language AL-log, used for mining multi-level association rules.

In this paper, we present a new semantic subgroup discovery system named Hedwig, which searches for subgroups with descriptions constructed from the given ontological vocabulary (including any provided binary relations). The traversal of the search space is effectively guided by the hierarchical structure of the ontology. The most relevant related work in exploiting ontologies in real-life data mining tasks is by Trajkovski et al. [13] who used the gene ontology to find enriched gene sets from microarray data, and by akova et al. [14] who used an ontology of Computer Aided Design elements and structures to find frequent design patterns.

In this paper, we present the results of applying the Hedwig system to get insight into a vast amount of news articles collected in last two years as part of the 7FP EU projects FIRST and FOC. We seek for insight in the financial domain; more specifically we investigate the vocabulary related to the European sovereign debt crisis used in news articles and financial blogs. We investigate the relationship between the financial market perception of a financial entity and the articles mentioning the financial entity. As a measure of market perception, we use the credit default swap (CDS) price. In essence, CDS is insurance for country bonds and reflects the market expectation that the issuer will default. The higher the CDS price, the more likely it is that that country will be unable to repay its debt [15]. Portugal is the focus of our investigation as an example of a financially troubled country.

Gamberger et al. [16] employed SD techniques on a related problem. They have induced indicators of systemic banking crises by looking at past crises in the period 1976-2007. Rather than looking at news articles and relating them to the CDS prices, they used 105 publicly available financial indicators. Their main result is that demographic indicators are the most important: the percentage of the active population in connection to the annual percentage of money growth and the male life expectancy are especially crucial.

The main contributions of this paper are the new semantic data mining system named Hedwig, which is presented with its premiere application in understanding financial news, and the extensive data acquisition pipeline that was used for collecting the data. Another contribution is the first insights into the relationship between the European sovereign debt crisis vocabulary and the CDS price trends.

The paper is structured as follows. Section 2 describes the developed Hedwig semantic SD system. Section 3 describes the data acquisition and cleaning pipeline, while Section 4 describes the data preparation stage, the experimental setup and the results. Section 5 gives directions for further work and concludes the paper.

2 Methodology

This section describes the newly developed semantic subgroup discovery system Hedwig. Compared to standard subgroup discovery algorithms, Hedwig uses domain ontologies to guide the search space and formulate generalized hypothesis. Existing semantic subgroup discovery algorithms are either specialized for a specific domain [13] or adapted from systems that do not take into the account the hierarchical structure of background knowledge [6]. Hedwig overcomes these limitations as it is designed to be a general purpose semantic subgroup discovery system.

Semantic subgroup discovery, as addressed by the Hedwig system, results in relational descriptive rules, using training examples in RDF triples form and using several ontologies as background knowledge used. As an illustration, take three simplified ontologies illustrated in Figure 1, as sample ontologies which could be used in mining financial data.

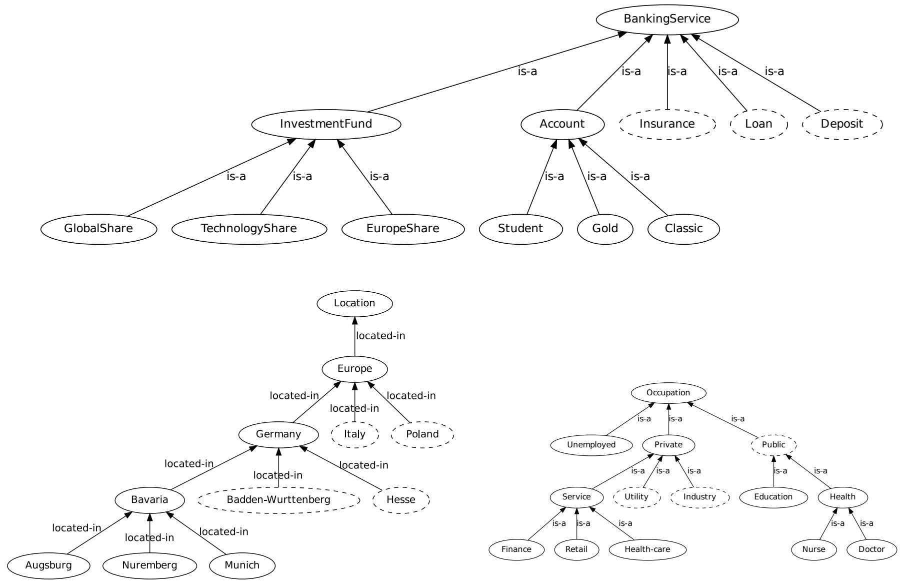


Fig. 1. The ontologies of banking services, locations and occupations. Concepts with omitted sub-concepts are drawn with a dashed line.

Formally, the semantic data mining task addressed in this paper is defined as follows.

Given:

– The empirical data in the form of a set of training examples expressed as RDF triples,
– Domain knowledge in the form of ontologies (one or more), and
– An object-to-ontology mapping which associates each object from the RDF triplets with appropriate ontological concepts.

Find:

– A hypothesis (a predictive model or a set of descriptive patterns), expressed by domain ontology terms, explaining the given empirical data.

Subgroup describing rules are first-order logical expressions. Take the following rule, used to explain the format of induced subgroup describing rules.

`Max(X) ← Country(X), Before(X,Y), comp_NESTLE_S_A(Y). [50, 10]`

where variables `X`, `Y` represent sets of input instances. Note the convention that lowercase predicates (e.g., `comp_NESTLE_S_A`) represent specific instances (appearing in the leaves of the ontology), while capitalized predicates represent classes (appearing at higher hierarchy levels of the ontology), i.e., sets of specific instances (e.g., predicate `Country` subsumes instances like `cou_Portugal` or

```
function induce():
    rules = [default_rule]
    while improvement(rules):
        foreach rule in rules:
            rules.extend(specialize(rule))
        rules = best(rules, N)
    return rules

function specialize(rule):
    specializations = []
    foreach predicate in eligible(rule.predicates):
        # Specialize by traversing the subClassOf hierarchy
        for subclass in subclasses(predicate):
            new_rule = rule.swap(predicate, subclass)
            if can_specialize(new_rule):
                specializations = specializations.add(new_rule)
    if rule != default_rule:
        # Specialize by adding a new unary predicate to the rule
        new_predicate = next_non_ancestor(eligible(rule.predicates))
        new_rule = rule.append(new_predicate)
        if can_specialize(new_rule):
            specializations.add(new_rule)
    if rule.predicates.last().arity == 1:
        # Specialize by adding new binary predicates
        specializations.extend(add_binary_predicate(rule))
    return specializations
```

Fig. 2. Pseudo code of the Hedwig semantic SD algorithm

`cou_Slovenia`). The above rule is interpreted as follows. Let `Max(X)` denote a local maximum of credit default swap (CDS), which needs to be related with the information available in the extracted features of news articles at time point `X`. The countries `Country(X)`, which were frequently mentioned in articles on day `X` that is followed by `Y` in which the Nestle company was frequently mentioned. This rule condition is true for 50 input instances, 10 of which are of target class `Max`. The two numbers refer to coverage (the number of instances for which the rule body is true) and support (the number of instances for which both the rule head and body are true), respectively.

In order to search for interesting subgroups, we employed the algorithm described in Figure 2. The Hedwig system, which implements this algorithm, supports ontologies and examples to be loaded as a collection of RDF triples (a graph). The system automatically parses the RDF graph for the `subClassOf` hierarchy, as well as any other user-defined binary relations. Hedwig also defines a namespace of classes and relations for specifying the training examples to which the input must adhere.

The algorithm uses beam search, where the beam contains the best N rules found so far. The search starts with the default rule which covers all input examples. In every iteration of the search, each rule from the beam is specialized via one of the three operations:

1. Replace the rules predicate with a predicate that is a sub-class of the previous one, e.g., `City(X)` is specialized to `Capital(X)`.
2. Append a new unary predicate to the rule, e.g., `Max(X)` $\leftarrow$ `City(X)` is specialized to `Max(X)` $\leftarrow$ `City(X), Company(X)`.
3. Append a new binary predicate, thus introducing a new existentially quantified variable, e.g.: `Max(X)` $\leftarrow$ `City(X)` is specialized to `Max(X)` $\leftarrow$ `City(X), Before(X,Y)`.[1]

Rule induction via specializations is a well-established way of inducing rules, since every specialization either maintains or reduces the current number of covered examples. A rule will not be specialized once its coverage is zero or falls below some predetermined threshold. After the specialization step is applied to each rule in the beam, a new selection of the best scoring N rules is made. If no improvement is made to the collection of rules, the search is stopped. In principle, our procedure supports any rule scoring function. Currently we implemented the popular SD scoring functions WRAcc [17], χ^2 for discrete target classes [18], and Z-score for ranked examples [19].

3 Data Acquisition and Cleaning

In this section, we present the data acquisition pipeline by describing each of its components.

The pipeline consists of several technologies that interoperate to achieve the desired goal, i.e., preparing the data for further analysis. It is responsible for acquiring unstructured data from several data sources, preparing it for the analysis, and brokering it to the appropriate analytical components. Our data acquisition pipeline is running continuously (since October 24, 2011), polling the Web and proprietary APIs for recent content, turning it into a stream of preprocessed text documents.

The news articles and web blogs are collected from 175 web sites and 2,600 RSS feeds, intentionally selected to have a strong bias for finance. We collect data from the main news providers and aggregators (like yahoo.com, dailymail.co.uk, nytimes.com, bbc.co.uk, wsj.com) and also from the main financial blogs (like zerohedge.com). The hundred most productive web sites account for 85% of collected documents. The fifty most productive domains with their average document production per day are displayed in Figure 3.

In the period from October 24, 2011 to March 31, 2013, 8,703,895 documents were collected and processed. On an average work day, about 18,000 articles are

[1] Note that variable `Y` needs to be 'consumed' by a literal to be conjunctively added to this clause in the next step of rule refinement.

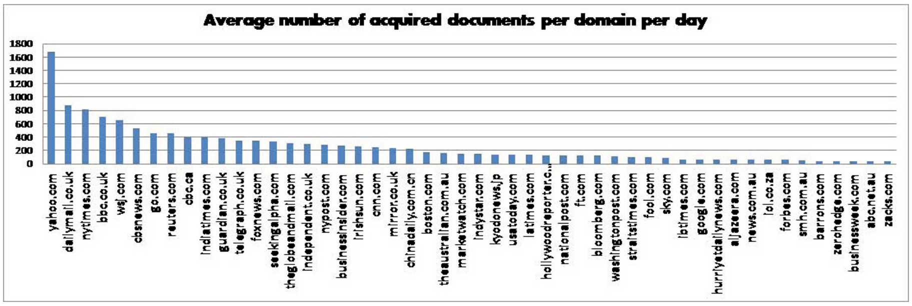


Fig. 3. The average number of acquired documents per domain per day for the fifty most productive domains. The hundred most productive web sites account for 85% of our acquired documents.

collected. The number of collected articles is substantially lower during weekends; around 10,000 per weekend day. Holidays are also characterized by a lower number of documents. The number of collected documents per day is presented in Figure 4.

When dealing with official news streams, some pre-processing steps can be avoid-ed. Official news is provided in a semi-structured fashion such that titles, publication dates, and other metadata are clearly indicated. Furthermore, named entities (i.e., company names and stock symbols) are identified in texts and article bodies are provided in a raw textual format without any boilerplate (i.e., undesired content such as advertisements, copyright notices, navigation elements, and recommendations).

Content from blogs, forums, and other Web content, however, is not immediately ready to be processed by the text analysis methods. Web pages contain a lot of noise that needs to be identified and removed before the content can be analyzed. For this reason, we have developed DacqPipe (or Dacq), a data acquisition and pre-processing pipeline. Dacq consists of (i) data acquisition components, (ii) data clean-ing components, (iii) natural-language preprocessing components, (iv) semantic anno-tation components, and (v) ZeroMQ emitter components.

The data acquisition components are mainly RSS readers that poll for data in parallel. One RSS reader is instantiated for each Web site of interest. The RSS sources, corresponding to a particular Web site, are polled one after another by the same RSS reader to prevent the servers from rejecting requests due to concurrency. An RSS reader, after it has collected a new set of documents from an RSS source, dispatches the data to one of several processing pipelines. The pipeline is chosen according to its current load size (load balancing). A processing pipeline consists of a boilerplate remover, duplicate detector, language detector, sentence splitter, tokenizer, part-of-speech tagger, lemmatizer, stop-word detector and a semantic annotator. Some of the components are custom-made while other use the functionality available from the OpenNLP library . Each pipeline component is described in more detail below.

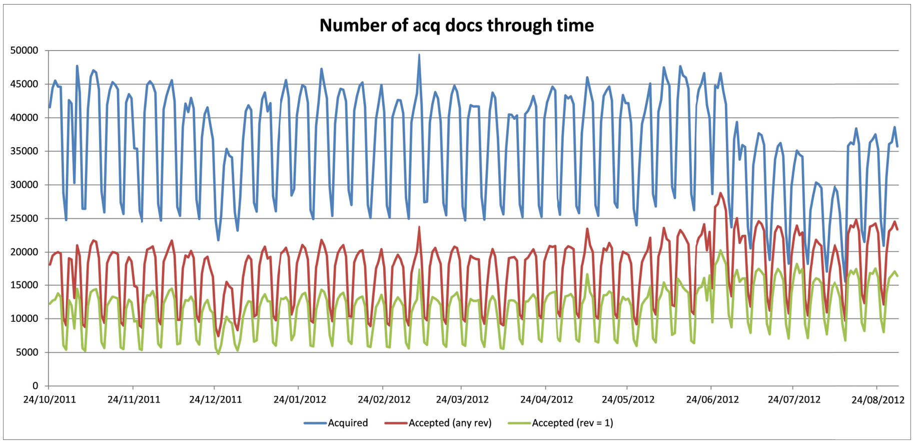


Fig. 4. The number of acquired documents per day. The top line represents the number of all acquired documents. The bottom line represents the documents that our system sees for the first time and the middle line represents the revisions of already acquired documents.

– *Boilerplate Remover.* Extracting meaningful content from Web pages presents a challenging problem. Our setting focuses on content extraction from streams of HTML documents. The developed infrastructure converts continuously acquired HTML documents into a stream of plain text documents. Our novel content extraction algorithm is efficient, unsupervised, and language-independent. The information extraction approach is based on the observation that HTML documents from the same source normally share a common template. The core of the proposed content extraction algorithm is a simple data structure called URL Tree. The performance of the algorithm was evaluated in a stream setting on a time-stamped semi-automatically annotated dataset which was made publicly available.
– *Duplicate Detector.* News aggregators are websites that aggregate web content such as news articles in one location for easy viewing. They cause articles to appear on the web with many different URLs pointing to it. To have a concise dataset of unique articles, we developed a duplicate detector that is able to see if the document was already acquired or not.
– *Language Detector.* By using a machine learning model, it detects the language and discards all the documents that are detected to be non-English. The model is trained on a large multilingual set of documents. The basic features for the model are frequencies of two consecutive words.
– *Sentence Splitter.* Splits the text into sentences. The result is the input to the part-of-speech tagger. We use the OpenNLP implementation of the Sentence splitter.

- *Tokenizer.* Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. In our pipeline, we use our own implementation of the tokenizer, which supports the Unicode character set and is based on rules.
- *Part-of-speech Tagger.* The Part of Speech (POS) Tagger marks tokens with their corresponding word type (e.g., noun, verb, proposition) based on the token itself and the context of the token. A token might have multiple POS tags depending on the token and the context. The part-of-speech tagger from the OpenNLP library is used.
- *Lemmatizer.* Lemmatization is the process of finding the normalized forms of words appearing in text. It is a useful preprocessing step for a number of language engineering and text mining tasks, and especially important for languages with rich inflectional morphology. In our data acquisition pipeline, we use LemmaGen [20] for lemmatization, which is the most efficient publicly available lemmatizer trained on large lexicons of multiple languages, whose learning engine can be retrained to effectively generate lemmatizers of other languages. We lemmatize to English.
- *Stop-word detector.* In automated text processing, stop words are words that do not carry semantic meaning. In our data acquisition pipeline, stop words are detected and annotated.
- *Semantic annotator.* Each entity has associated gazetteers; gazetteers are rules describing the entity in text. For example, “The United States of America” can appear in text as “USA”, “US”, “The United States”, and so on. The rules include capitalization, lemmatization, POS tag constraints, must-contain constraints (another gazetteer must be detected in the document or in the sentence) and followed-by constraints.

4 Financial Use Case

First, this section presents the data and the data preparation stage needed to apply the proposed methodology. Three sources of data were used: texts from news and blogs, CDS prices and a domain ontology. Finally, this section present the experimental results achieved by applying subgroup discovery on the prepared data.

We started from a large database of annotated news articles (over 8 million), which were acquired using the data acquisition pipeline presented in the previous section. We considered articles collected over eighteen months period from October 24, 2011 to January 13, 2013. Among other properties of each article (e.g., title and URL), the most important ones for our task are the information about which entities from a pre-defined European Sovereign Debt vocabulary appear in the given article (e.g., entities like “Portugal” or “Angela Merkel” or “austerity”). These entities (counting over 6,000) are part of a larger domain ontology which consists of several class hierarchies, e.g., the Euro crisis vocabulary, companies and banks, and geographical data.

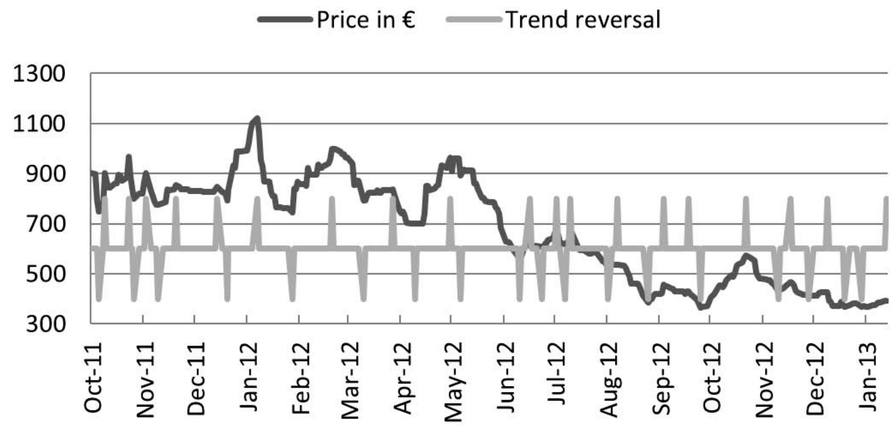


Fig. 5. Portugal CDS prices and trend reversals between October 2011 and January 2013. Upward spikes indicate local maxima, while downward spikes indicate local minima.

We decided to focus our experiments on Portugal, as it is representative and was a financially troubled country in the analyzed period. Therefore the news articles were filtered to include only the articles mentioning Portugal. The preparation stage consisted of two steps. The first step involved counting the number of times Portugal occurs together with every other entity of interest for each day of the collected history of articles. The second step involved selecting only the significant co-occurrences as example features. Each day represents one learning example and each example is described by the presence or absence of a certain entity that co-occurred with Portugal on that day. To filter out uninformative entities, we kept only the entities with a co-occurrence frequency at least 1.5 times greater than the average co-occurrence frequency over all days.

The target attribute for each example (day) was computed from the CDS prices of Portugal and has three possible values that indicate the significant local extremes in the CDS price timelines: 'max' or 'min' if the local extreme was reached, respectively, or 'steady' if there was no change in the trend (Figure 5). These steps yielded a dataset of 337 examples, each with an average of 282 features (ranging between 35 and 761).

The processed news and blogs articles, the CDS local extremes and the domain ontology were encoded as a set of RDF triples which were input to the Hedwig system.

The financial ontology which we actually used in the experiments is illustrated in Figure 6. The ontology has three main branches: financial entities, geographical entities and a specialized vocabulary of the European sovereign debt crisis. Some parts of the ontology were automatically induced by reusing various data sources, while other parts, like the vocabulary, were constructed manually.

Each entity in the ontology is equipped with a gazetteer. The gazetteer contains lexical knowledge about the possible forms in which the entity occurs in texts. This knowledge is used by the entity recognition engine which is attached to the data acquisition pipeline. Note that the gazetteers are initially built automatically in the ontology construction process. This approach to entity recognition is prone to errors due to homographs, i.e., words that are spelled the same but have different meanings. This is especially prominent for acronyms and stock symbols. To improve the entity recognition process and to reduce the noise in the stream of discovered entities, we have performed several semi-automated ontology refinement iterations.

We used the IDMS database and MSN Money[2] to grow the ontology from a list of seed stock indices to its constituents (stocks) and further on to the companies that issue these stocks. This resulted in to 2019 financial entities (like banks, companies, investment funds, stocks and stock indexes). The geographical part of the ontology was generated from GeoNames[3] (countries, cities, regions, etc). We selected 598 most important geographical entities and included them into the ontology. The specialized vocabulary of the European financial crisis (166 terms) was developed manually by using expert knowledge (Figure 6). The main protagonists of the crisis were taken from Wikipedia[4].

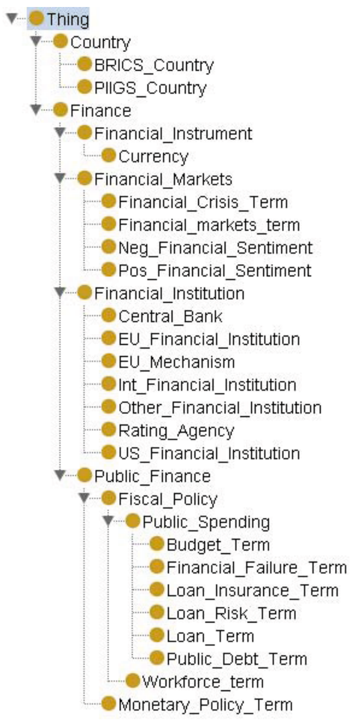


Fig. 6. The ontology that conceptualizes the European financial crisis vocabulary

In our experiment, we focused on finding subgroups for two target classes which represent trend reversals: the local maximum ('max') represents the date when the CDS price started to decrease and the local minimum ('min') the opposite. In both cases, we used the WRAcc subgroup discovery rule score, a beam width of 100, minimum coverage of 5 examples and the maximum number of predicates per rule of 6.

[2] http://money.msn.com/

[3] http://www.geonames.org/

[4] http://en.wikipedia.org/wiki/List_of_protagonists:_European_sovereign-debt_crisis

For the case of CDS price reaching the maximum (target class 'max'), the best scoring subgroup was:

```
Max(X) ← reg_Western_Europe(X), Angela_Merkel(X),
   glo_austerity(X), glo_recession(X). [28, 7]
```

For the case of CDS price reaching the minimum (target class 'min'), the best scoring subgroup was:

```
Min(X) ← Index(X), comp_GALP_ENERGIA(X), Loan_Term(X),
   glo_fiscal_stimulus(X). [43, 8]
```

The first rule indicates that Portugal CDS prices reaching a local maximum are characterized by increased frequency of the following entities co-occurring with Portugal: the Western Europe region, Angela Merkel, and the terms 'austerity' and 'recession'. We should point out that a local maximum in a country's CDS price indicates that from that day on, the market expectation that the country will default decreased. Conversely, the second rule tells us that when the CDS price reach a local minimum, we can notice an increased frequency of (stock) index terms, Portugal's corporation of natural and renewable energy companies (Galp Energia), loan terms and 'fiscal stimulus'. These results show that the higher the CDS prices, the more the sovereign debt vocabulary is used. When CDS prices are low, a more general financial terminology is used.

5 Conclusions

The newly developed semantic subgroup discovery system Hedwig was presented, which overcomes the limitations of existing semantic subgroup discovery systems. Compared to standard subgroup discovery, novelties of this paper are the exploitation of the ontology to generalize over the entities, while also using of the user-provided binary relations and using the `subClassOf` relation to guide the search procedure. We are currently performing a comprehensive study which should result in a comparison of the new system with the related work.

We employed Hedwig for analyzing news articles about Portugal during the last year and a half. Using co-occurrence frequencies of entities appearing together with Portugal, a domain ontology linking the entities into a formal hierarchy, and a history of Credit Default Swap (CDS) prices, we induced subgroups describing prominent entities appearing at times of CDS trend reversals (either upward or downward). The extracted subgroup descriptions give us a clear indication that news articles content indeed reflects the CDS prices. Having this information, we are encouraged to proceed with building a model for CDS trend reversal prediction. For this purpose, we plan to include additional information about the entities (e.g., TF-IDF weights) and extra-textual information (not only the pre-defined ontological entities) into the input data. Additionally, we will employ several classification algorithms and compare them.

Acknowledgments. This work was supported by the Slovenian Research Agency [grants P-103 and P-04431] and the EU projects "Large scale information extraction and integration infrastructure for supporting financial decision making" (FIRST, grant agreement 257928) and "Forecasting Financial Crises" (FOC, grant agreement 255987).

References

[1] Klösgen, W.: Explora: a multipattern and multistrategy discovery assistant. In: Advances in Knowledge Discovery and Data Mining, pp. 249–271. American Association for Artificial Intelligence, Menlo Park (1996)

[2] Wrobel, S.: An algorithm for multi-relational discovery of subgroups. In: Komorowski, J., Żytkow, J.M. (eds.) PKDD 1997. LNCS, vol. 1263, pp. 78–87. Springer, Heidelberg (1997)

[3] Muggleton, S. (ed.): Inductive Logic Programming. The APIC Series, vol. 38. Academic Press (1992)

[4] De Raedt, L.: Logical and Relational Learning. Springer, Heidelberg (2008)

[5] Džeroski, S., Lavrač, N. (eds.): Relational Data Mining. Springer, Berlin (2001)

[6] Vavpetič, A., Lavrač, N.: Semantic subgroup discovery systems and workflows in the SDM-Toolkit. Comput. J. 56(3), 304–320 (2013)

[7] Kietz, J.-U.: Learnability of description logic programs. In: Matwin, S., Sammut, C. (eds.) ILP 2002. LNCS (LNAI), vol. 2583, pp. 117–132. Springer, Heidelberg (2003)

[8] Lehmann, J., Haase, C.: Ideal downward refinement in the $\mathcal{EL}$ description logic. In: De Raedt, L. (ed.) ILP 2009. LNCS, vol. 5989, pp. 73–87. Springer, Heidelberg (2010)

[9] Ławrynowicz, A., Potoniec, J.: Fr-ONT: An algorithm for frequent concept mining with formal ontologies. In: Kryszkiewicz, M., Rybinski, H., Skowron, A., Raś, Z.W. (eds.) ISMIS 2011. LNCS, vol. 6804, pp. 428–437. Springer, Heidelberg (2011)

[10] Berendt, B., Hotho, A., Stumme, G.: Towards semantic web mining. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 264–278. Springer, Heidelberg (2002)

[11] Lisi, F.A., Malerba, D.: Inducing multi-level association rules from multiple relations. Machine Learning 55, 175–210 (2004), 10.1023/B:MACH.0000023151.65011.a3

[12] Lisi, F.A., Esposito, F.: Mining the semantic web: A logic-based methodology. In: Hacid, M.-S., Murray, N.V., Raś, Z.W., Tsumoto, S. (eds.) ISMIS 2005. LNCS (LNAI), vol. 3488, pp. 102–111. Springer, Heidelberg (2005)

[13] Trajkovski, I., Železný, F., Lavrač, N., Tolar, J.: Learning relational descriptions of differentially expressed gene groups. IEEE Transactions on Systems, Man, and Cybernetics, Part C 38(1), 16–25 (2008)

[14] Žáková, M., Železný, F., Garcia-Sedano, J.A., Tissot, C.M., Lavrač, N., Křemen, P., Molina, J.: Relational data mining applied to virtual engineering of product designs. In: Muggleton, S.H., Otero, R., Tamaddoni-Nezhad, A. (eds.) ILP 2006. LNCS (LNAI), vol. 4455, pp. 439–453. Springer, Heidelberg (2007)

[15] Hull, J., Predescu-Vasvari, M., White, A., Rotman, J.L.: The relationship between credit default swap spreads, bond yields, and credit rating announcements (2002)

[16] Gamberger, D., Lučanin, D., Šmuc, T.: Descriptive modeling of systemic banking crises. In: Ganascia, J.-G., Lenca, P., Petit, J.-M. (eds.) DS 2012. LNCS, vol. 7569, pp. 67–80. Springer, Heidelberg (2012)
[17] Lavrač, N., Kavšek, B., Flach, P., Todorovski, L.: Subgroup discovery with CN2-SD. Journal of Machine Learning Research 5, 153–188 (2004)
[18] Shimada, K., Hirasawa, K., Hu, J.: Class association rule mining with chi-squared test using genetic network programming. In: IEEE International Conference on Systems, Man and Cybernetics, SMC 2006, vol. 6, pp. 5338–5344 (2006)
[19] DeGroot, M.H., Schervish, M.J.: Probability and Statistics, ch. 8, 9. Addison-Wesley (2002)
[20] Juršič, M., Mozetič, I., Erjavec, T., Lavrač, N.: Lemmagen: Multilingual lemmatisation with induced ripple-down rules. J. UCS 16(9), 1190–1214 (2010)

Polynomial Delay and Space Discovery of Connected and Acyclic Sub-hypergraphs in a Hypergraph

Kunihiro Wasa[1], Takeaki Uno[2], Kouichi Hirata[3], and Hiroki Arimura[1]

[1] Graduate School of IST, Hokkaido University, Sapporo, Japan
`{wasa,arim}@ist.hokudai.ac.jp`
[2] National Institute of Informatics, Tokyo, Japan
`uno@nii.jp`
[3] Dept. of Artificial Intelligence, Kyushu Institute of Technology, Iizuka, Japan
`hirata@ai.kyutech.ac.jp`

Abstract. In this paper, we study the problem of finding all tree-like substructure contained in a hypergraph, with potential applications to substructure mining from relational data. We employ the class of connected and Berge acyclic sub-hypergraphs as definition of tree-like substructures, which is the most restricted notion of acyclicities for hypergraphs. Then, we present an efficient depth-first algorithm that finds all connected and Berge acyclic sub-hypergraphs S in a hypergraph $\mathcal{H}$ with m hyperedges and n vertices in $O(nm^2)$ time per solution (delay) using $O(N)$ space, where $N = ||\mathcal{H}||$ is the total input size. To achieve efficient enumeration, we use the notion of the maximum border set. This result gives the first polynomial delay and time algorithm for enumeration of connected and Berge-acyclic sub-hypergraphs. We also present an incremental enumeration algorithm that finds all solutions S in $O(\Delta MB(S)\tau(m)) = O(rd\cdot\tau(m))$ delay using $O(N)$ space and preprocessing, whose delay depends only on the difference of solutions, where S is the enumerated sub-hypergraph, $\Delta MB(S)$ is the number of newly added hyperedges to the maximum border of S, r and d are the rank and degree of $\mathcal{H}$, respectively, and $\tau(m) = ((\log\log m)^2/\log\log\log m)$.

1 Introduction

In data mining, it is a well-studied problem to discover all interesting substructures of a given discrete structure under various notions of substructures. Particularly, examples of such substructure discovery are frequent itemset mining [25, 28, 29], sequence mining [1, 2, 19], trees and graph mining [3, 13, 14, 27], and kernel-like similarity computation [16] to name a few.

In this paper, we study the problem of enumerating all connected and acyclic sub-hypergraphs contained in an input hypergraph for the notions of acyclicity, called Berge-acyclicity [6], which is at the bottom of hierarchy of acyclicities given by Fagin [9]. A *hypergraph* is a pair $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ of a collection $\mathcal{V}$ of vertices and a collection $\mathcal{E}$ of hyperedges (See Fig. 1 for example), where a hyperedge is any finite set $e \subseteq \mathcal{V}$ of vertices. Essentially, a hypergraph is a representation of

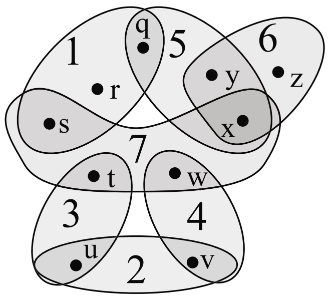


Fig. 1. An example of a hypergraph $\mathcal{H}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ with the vertex set $\mathcal{V}_1 = \{p, q, r, \dots, x, y, z\}$ and the hyperedge set $\mathcal{E}_1 = \{1, 2, 3, 4, 5, 6, 7\}$

a *set collection* $\mathcal{E}$ consisting of groups of objects taken from a common universe $\mathcal{V}$. For example, the followings are examples of such set collections: transaction databases, author groups in bibliographic data, co-citation networks in social networks, and interaction graphs for genes and proteins in bioinformatics [17]. In such networks, discovery of substructures such as connected components, connected subtrees, cliques, quasi-cliques, and dense subgraphs have been extensively studied in the context of network mining [17, 21, 24].

Recently, discovery of *subtrees* in a graph, which are connected and acyclic edge subsets, attract much attention [10, 26]. By generalizing the notion of subtrees in a graph to a hypergraph, we consider discovery of the class of *connected acyclic sub-hypergraphs* appearing in a hypergraph. Particularly, among several definitions of acyclicities in a hypergraph, we employ the most restricted one, called *Berge-acyclicity* [6], where a hypergraph is Berge-acyclic if and only if it contains no cycle of hyperedges. For example, in the example of Fig. 1, the hyperedge subset $S = \{1, 3, 4, 7\}$ is connected Berge-acyclic sub-hypergraph. It is known that Berge-acyclicity locates the bottom of the degrees among α-, β-, and γ-acyclicities [9]. Now, our goal is to devise an efficient algorithm for the connected, Berge-acyclic sub-hypergraph mining problem in *polynomial delay* (time per solution) and *polynomial space*, which means that the algorithm achieves high-throughput and small memory footprint computation in large scale applications.

Main results: Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be an input hypergraph consisting of n vertices and m hyperedges. Then, we present efficient depth-first algorithms BergeMine that finds all connected Berge-acyclic sub-hypergraphs S contained in $\mathcal{H}$ without duplicates in $O(m^2 n)$ delay and $O(N)$ words of space (Theorem 1). To achieve polynomial delay and space complexity, our algorithm searches for all solutions in the depth-first manner on a tree-shaped search space without using any extra memory for table-lookup. This search space is designed based on a characterization of Berge-acyclic sub-hypergraphs given by us, with which we proposed an efficient and complete pruning strategy.

Next, we present the modified algorithm, called FastBergeMine, that finds all S in $O(\Delta MB(S)\tau(m)) = O(rd \cdot \tau(m))$ delay using $O(N)$ space and preprocessing, where S is the enumerated sub-hypergraph, $\Delta MB(S)$ is the number of newly added hyperedges to the maximum border of S, r and d are the rank and degree of $\mathcal{H}$, respectively, and $\tau(m) = ((\log\log m)^2/\log\log\log m)$. The algorithm uses incremental computation of the maximum border set. Since it has the delay that depends only on the size of each discovered subset S and its neighbors $N(S)$, it will be more efficient for the large inputs in the real world.

Related work: For the class of α-acyclic sub-hypergraphs [9], Hirata *et al.* [12] presented an efficient algorithm that finds one of the maximal connected and acyclic sub-hypergraphs in an input hypergraph in linear time in the total input size. Extending this work, Daigo and Hirata [8] presented a polynomial delay and space algorithm that finds all connected and acyclic sub-hypergraphs in an input hypergraph. For the class of Berge-acyclic sub-hypergraphs, Lovász [18] showed a polynomial time algorithm that finds one of the maximal connected and Berge-acyclic sub-hypergraphs in an input hypergraph.

As closely related work, Ferreira *et al.* [11] presented an efficient algorithm for finding all distinct subtrees of size k in an input graph in $O(k)$ time (time per solution) and space, and Wasa *et al.* [26] the improved version in constant delay when an input is a tree. However, their approaches cannot be directly applicable to our problem.

In the case that the maximum size of hyperedges, the *rank*, is restricted to two, the problem coincides to the well-known spanning tree problem for undirected graphs. For the problem, Tarjan and Read [23] first presented an $O(ns)$ time and $O(n)$ space algorithm in 1960's, where n is the number of edges in G. Recently, Shioura, Tamura, and Uno [20] presented $O(n+s)$ time and $O(n)$ space algorithm. Unfortunately, it is not easy to extend the algorithms for spanning tree enumeration to subtree enumeration.

Organization of this paper: In Sec. 2, we give basic definitions and notations on hypergraphs and our data mining problem. In Sec. 3, we present the basic depth-first algorithm BergeMine for the problem. in Sec. 4, we present the modified version of the algorithm, FastBergeMine, using incremental computation. Finally, in Sec. 5, we conclude.

2 Preliminaries

In this section, we give the definitions and notations for hypergraphs. For the definitions not found here, please consult appropriate textbooks (e.g., [6,7]). For a set A, we denote by $|A|$ the cardinality of A. For a collection $\mathcal{X} \subseteq 2^A$ of subsets of A, $||\mathcal{X}|| = \sum_{S\in\mathcal{X}} |S|$ denotes the *total size* of $\mathcal{X}$.

2.1 Hypergraphs

Intuitively, a hyper graph is a structure defined by a set collection $\mathcal{E} \subseteq 2^{\mathcal{V}}$ over some finite domain $\mathcal{V}$ of objects. Formally, a *hypergraph* over a set of vertices $\mathcal{V}$ is any pair $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ consists of the following components:

- A set of *vertices* $\mathcal{V} = \mathcal{V}(\mathcal{H}) = \{1, \ldots, n\}$, $n \geq 0$, and
- A set of *hyperedges* $\mathcal{E} = \mathcal{E}(\mathcal{H}) = \{e_1, \ldots, e_m\}$, $m \geq 0$,

where for every $1 \leq i \leq m$, e_i is a subset of $\mathcal{V}$, called a hyperedge, and its index i is called the *edge ID* of e_i. Since $\mathcal{E} \subseteq 2^{\mathcal{V}}$, the number m of hyperedges can be exponential in n. The *total size* of $\mathcal{H}$, denoted by $||\mathcal{H}|| = N$, is the sum of the sizes of its hyperedges, that is, $||\mathcal{H}|| = ||\mathcal{E}|| = \sum_{e \in \mathcal{E}} |e|$. By definition, $||\mathcal{E}|| \leq O(mn)$.

For vertex x and hyperedges e, f, we say that e is *incident to* x (or e *contains* x) if $x \in e$ holds, and that e is a *neighbor* of f if $f \cap e \neq \emptyset$ holds. Then, $N(x) = \{\, f \in \mathcal{E} \mid x \in f \,\}$ is the set of all hyperedges incident to x, and $NE(e) = \{\, f \in \mathcal{E} \mid e \cap f \neq \emptyset \,\}$. Then, the *rank* of $\mathcal{H}$, denoted by $r = \mathit{rank}(\mathcal{H}) = \max_{f \in \mathcal{E}} |f|$, is the maximum size of its hyperedges. The *degree* of $\mathcal{H}$, denoted by $d = \mathit{deg}(\mathcal{H}) = \max_{x \in \mathcal{V}} |N(x)|$, is the maximum number of incident edges. The *hyperedge degree* of $\mathcal{H}$, denoted by $D = \mathit{hyperedge\text{-}deg}(\mathcal{H}) = \max_{e \in \mathcal{E}} |NE(e)|$, is the maximum number of the neighbors. Clearly, we see that $r = \mathit{rank}(\mathcal{H}) \leq n$, $d = \mathit{deg}(\mathcal{H}) \leq m$, and $D = \mathit{hyperedge\text{-}deg}(\mathcal{H}) \leq \min\{m, rd\}$.

In this paper, a *sub-hypergraph* of $\mathcal{H}$ is just a subset $S = \{e_{i_1}, \ldots, e_{i_k}\} \subseteq \mathcal{E}$ $(k \leq m)$ of hyperedges of $\mathcal{H}$.[1] We use the terms a hyperedge subset and a sub-hypergraphs interchangeability in what follows. Actually, a subset S induces a hypergraph $\mathcal{H}[S] = (S, \mathcal{E}[S])$, where $\mathcal{E}[S] = \{\, e \in \mathcal{E} \mid e \subseteq S \,\}$. The *neighbor set* of S is $N(S) = \{f \in \mathcal{E} \setminus S \mid e \in S, e \cap f \neq \emptyset\}$. Clearly, $||S + N(S)|| \leq ||\mathcal{E}|| = O(mn)$.

In what follows, we refer to vertices as $x, y, \ldots$, hyperedges as $e, f, \ldots$, and hyperedge subsets as $S, T, \ldots$, possibly subscripted. For convenience, we often represent a set of elements $\{a, b, c\}$ by a juxtaposition abc if it is clear from context.

Example 1. In Fig. 1, we show an example of a hypergraph $\mathcal{H}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ consisting of eleven vertices $\mathcal{V}_1 = \{p, q, r, \ldots, x, y, z\}$ and seven hyperedges $\mathcal{E}_1 = \{e_1, e_2, \ldots, e_7\}$ such that $e_1 = qrs, e_2 = uv, e_3 = tu, e_4 = vw, e_5 = qxy, e_6 = xyz$, and $e_7 = stwx$.

2.2 Connected and Berge-Acyclic Sub-hypergraphs

A *path* between hyperedges e and $f \in \mathcal{E}$ in $S \subseteq \mathcal{E}$ is a sequence $\pi = (e_1 = e, e_2, \ldots, e_k = f)$ $(k \geq 1)$ of hyperedges that satisfies the condition $e_i \cap e_{i+1} \neq \emptyset$ for every $1 \leq i \leq k - 1$.

Definition 1. A hyperedge subset S is *connected* if any pair of hyperedges e and f has some path between them in S.

[1] The definition of a sub-hypergraph in this paper is also referred to as a *partial hypergraph* in literatures.

By definition, the empty set and singleton set of hyperedges are connected. We can easily test the connectivity of a given subset S in $O(||S||)$ time.

Example 2. In the hypergraph $\mathcal{H}_1$ in Example 1, the subsets $S_1 = 1567 = \{1, 5, 6, 7\}$ and $S_2 = 1347$ are connected. On the other hand, the subset $S_3 = 135$ is not connected since there is no path between the edges 1 and 3, and also the edges 5 and 3.

Definition 2 ([6]). *In a hypergraph $\mathcal{H}$, a* Berge-cycle *(or simply a* cycle*) of length k is a sequence $\pi = (e_1, x_1, \ldots, e_k, x_k)$ $(k \geq 2)$ that satisfies the following conditions (i)–(iii):*

(i) $e_1, \ldots, e_k$ are mutually distinct hyperedges.
(ii) $x_1, \ldots, x_k$ are mutually distinct vertices.
(iii) For each $1 \leq i \leq k-1$, $x_i \in e_i \cap e_{i+1}$ holds, and $x_k \in e_k \cap e_1$ holds.

In the above definition, we say that the set $\{e_1, \ldots, e_k\}$ of hyperedges *forms a Berge-cycle* Intuitively, a Berge-cycle is a path of length more than or equal to two that starts from some hyperedge and returns to the start.

Example 3. In the hypergraph $\mathcal{H}_1$ in Example 1, the hyperedge subset $S_4 = 157$ forms a Berge-cycle $\pi_4 = (1, q, 5, x, 7, s)$ of length three. From Lemma 1, we also see that the pair of hyperedges $S_5 = 56$ forms a Berge-cycle $\pi_5 = \langle 5, x, 6, y \rangle$ of length two since hyperedges 5 and 6 share common vertices x and y.

From the construction of minimum length cycle S_5 in the above example, we have the following lemma, which is well-known providing a fundamental property of Berge-acyclicity.

Lemma 1 (Berge [6]). *If two hyperedges e and f contain mutually distinct vertices x and y in common, i.e., $x, y \in e \cap f$, then they form a Berge-cycle.*

Proof. Take a path $\pi = (e, x, f, y)$ as a Berge-cycle. ■

Definition 3 (Berge-acyclic subgraph [6]). *A sub-hypergraph S is* Berge-acyclic *if it contains no Berge-cycles.*

By definition, the empty set and any singleton sets of hyperedges are Berge-acyclic. From the next lemma, Berge-acyclicity is closed under subsets.

Lemma 2. *If a non-empty subset S is Berge-acyclic, then any subset S' $(S' \subseteq S)$ is also Berge-acyclic.*

From Lemma 1 above, we see that Berge-acyclicity has strong restriction compared to other notions of hypergraph acyclicities. Actually, there is a hierarchy of acyclicities for hypergraphs, called the *degrees of acyclicities* of Fagin [9], that consists of α-acyclicity, β-acyclicity, γ-acyclicity, and Berge-acyclicities. In this hierarchy, α-acyclicity is most general, while Berge-acyclicity is most restricted.

In what follows, we denote by $\mathcal{AC} = \mathcal{AC}(\mathcal{H})$ the class of *all connected, and Berge-acyclic sub-hypergraphs* in an input hypergraph $\mathcal{H}$. Now, we state our data mining problem.

$$A_1 = \begin{array}{c|cccccccccc} & q & r & s & t & u & v & w & x & y & z \\ \hline 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 7 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}$$

Fig. 2. The incident matrix A_1 of the hypergraph $\mathcal{H}_1$ in Fig. 1, where each row indicates a hyperedge and each column a vertex

Definition 4 (Connected and Berge-acyclic sub-hypergraph mining problem in a hypergraph). *Given an input hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, the task is to find all connected, and Berge-acyclic sub-hypergraphs $S \subseteq \mathcal{E}$ in $\mathcal{H}$ belonging to the class $\mathcal{AC}(\mathcal{H})$ without duplicates.*

Example 4. Consider the hypergraph $\mathcal{H}_1$ in Example 1 again. Then, the subset $S_1 = 1567$ is not a connected and Berge-acyclic subset in $\mathcal{AC}(\mathcal{H}_1)$ because it is connected but cyclic. On the other hand, the subset $S_2 = 1347$ is a connected and Berge-acyclic subset in $\mathcal{AC}(\mathcal{H}_1)$.

The model of computation: An *enumeration algorithm* $\mathcal{A}$ receives an instance of size n and outputs all of m solutions without duplicates (See, e.g. [4]). For polynomials $p(\cdot,\cdot), q(\cdot), r(\cdot)$, $\mathcal{A}$ is of *output* $O(p(n,m))$*-time* if the total time of $\mathcal{A}$ is bounded by polynomial in n and m. $\mathcal{A}$ is of $O(q(n))$-delay using preprocessing $r(n)$ if the *delay*, which is the maximum computation time between two consecutive outputs, is bounded by $q(n)$ after preprocessing in $r(n)$ time. Clearly, if $\mathcal{A}$ is polynomial delay, A is also output polynomial time. $\mathcal{A}$ is of *polynomial space* if the maximum size of its working space is bounded by some polynomial $p(n)$.

2.3 Other Definitions and Properties

Leaves and connection counts: Let $S \subseteq \mathcal{E}$ be a subset of hyperedges, or a sub-hypergraph of $\mathcal{H}$. A hyperedge e *connects* S if the intersection $e \cap V(S)$ is not empty. Any vertex x in the intersection is called a *connection point*. Then, the *connection count* of e relative to S is defined by $cnt(e, S) = |e \cap V(S)| \geq 0$. In the next section, we give a characterization of connected and Berge-acyclic sub-hypergraphs using the connection count.

A *leaf* of a subset S is a hyperedge $e \in S$ such that $cnt(e, S - e) = 1$, that is, that has a single connection point in S except itself. Clearly, the empty subset $\emptyset$ has no leaf at all, and any singleton $S = \{e\}$ has the hyperedge e as its only leaf. We denote by $L(S)$ the *set of all leaves* of S. Actually, $L(S) = \{\, e \in S \mid cnt(e, S \setminus \{e\}) = 1 \,\}$.

Representation of hypergraphs: Using the incident relation $x \in e$, a hypergraph $\mathcal{H}$ with n vertices and m hyperedges can be represented as an $n \times m$ binary matrix

$A = (a_{i,j}) \in \{0,1\}^{m \times n}$, called the *incident matrix* of $\mathcal{H}$. For every $1 \le i \le n$ and $1 \le j \le m$, $a_{i,j} = 1$ if and only if $x_i \in e_j$ holds. In an incident matrix $A = (a_{i,j})$, each row $1 \le i \le n$ represents the incident set $N(x_i) \subseteq \mathcal{E}$, while each column $1 \le j \le m$ represents the corresponding hyperedge $e_j \subseteq \mathcal{V}$. In Fig. 2, we show an example of an incident matrix.

Data structure: In our algorithms in Sec. 3 and Sec. 4, we use a dynamic data structure $\mathcal{D}$ similar to the *DLX* (also known as "*Dancing Links*") data structure by Knuth [15] for dynamically maintaining a hyperedge subset S. Our data structure $\mathcal{D}$ stores the incident matrix of a set collection $D \subseteq \mathcal{E}$ in linear words of space in $||D||$ supporting the following operations: (i) retrieval of a hyperedge $e = e_i$ by an edge ID i, (ii) retrieval of the neighbor $N(x, D)$ by a vertex x, and (ii) insert/delete of elements to/from an edge or a neighbor set. Using dynamic predecessor dictionaries [7] (such as the hash table or the `map` collection of C++/STL or Java), we can execute the above operations in sublinear time $t = \tau(k)$, where we have $\tau(k) = \log k$ if we use ordinary binary tree, and $\tau(k) = O(((\log\log k)^2/\log\log\log k))$ for $k = \max\{n, m\}$ if we use the dynamic data structure of Beame and Fich [5]. The details are omitted here.

3 The Basic Algorithm

In this section, we show the basic version of our DFS mining algorithm BergeMine that finds all connected, and Berge-acyclic sub-hypergraphs in $\mathcal{H}$ in polynomial delay and space. In what follows, we write $S - e$ for $S' \setminus \{e\}$.

To devise efficient depth-first search algorithm, we need a systematic way to reduce the search for larger subsets to smaller subsets. The next lemma is essential to our algorithm.

Lemma 3. *Let $S' \subseteq \mathcal{E}$ is a subset such that $|S'| \ge 2$. If S' is connected and Berge-acyclic, then $cnt(e, S' - e) = 1$ holds for some $e \in S'$. Furthermore, $S = S' - e$ is connected and Berge-acyclic, too, and has size $|S| < |S'|$.*

Proof. We can show the lemma by induction on $|S'|$. If $|S'| = 2$, the claim is obvious since S' consists of two edges. Otherwise, assume that $|S'| > 2$. Since S' is connected, $cnt(e, S - e) \ge 1$ always holds for any $e \in S'$. Furthermore, if $cnt(e, S - e) \le 1$ holds for some $e \in S'$, then we are done. Therefore, we assume that $cnt(e, S - e) \ge 2$ holds for any $e \in S'$. Consider this case. Then, we split S' by removing e, and consider the connected components $S_1, \ldots, S_k$ of $S' - e$, where $k \ge 1$. There are two cases. (i) If e connects to some S_i at at least two points, then $S_i \cup \{e\}$, and thus S', immediately has a cycle, and we are done (ii) Otherwise, using induction hypothesis, we can show that there exists an edge f in some component, say S_1, such that $\{e, f\} \cup R$ forms a cycle for some $R \in \{\{e\}, S_1 - f, S_2, \ldots, S_k\}$ (details are omitted), and we are done. Hence, by contradiction, the lemma follows. ■

From Lemma 3 above, Starting from any connected and Berge-acyclic subhypergraph S with more than one edges, we can obtain a series of sub-hypergraphs

Algorithm 1. A basic algorithm BergeMine for mining all connected, Berge-acyclic sub-hypergraphs based on the reverse search

```
1: procedure BergeMine($\mathcal{H} = (\mathcal{V}, \mathcal{E})$: input hypergraph )
2:     BasicRec($\emptyset, \mathcal{H}$);
3: procedure BasicRec($S$: sub-hypergraph, $\mathcal{H}$: input hypergraph)
4:     Output $S$;
5:     $Border(S) \leftarrow \{ f \in (\mathcal{E}(\mathcal{H}) \setminus S) \mid cnt(f, S) = 1 \}$;
6:     for each $f \in Border(S)$ do                    ▷ Generation of children
7:         $S' \leftarrow S \cup \{f\}$;
8:         if $f = \max L(S')$ then
9:             BasicRec($S', \mathcal{H}$);
```

$\mathcal{R} = S_0 = S \supset S_1 \supset \cdots \supset S_\ell = \{e\}$ of length $\ell = |S| - 1 \geq 0$. Our DFS algorithm reverses this process by starting from any singleton set $\{e\}$, $e \in \mathcal{E}$, and by iteratively expanding the current subset $S \subseteq \mathcal{E}$ by adding new hyperedge $e \in \mathcal{E} \setminus S$ in a systematic manner using backtracking.

However, there is one problem with this approach. The above DFS search process may generate the same subset by exponentially many different paths. One easy way to avoid this duplication is to use table-lookup. When we discovered a new subset S, we lookup a hash table H to decide if $S \in H$. If so, we skip S, and otherwise, we output S and register it to H. This modification yields a polynomial delay, but exponential space mining algorithm.

We solve this problem by pruning of redundant path by careful design of the tree-shaped search space described as follows. Recall the previous key lemma, Lemma 3. In the lemma, the possible source of redundancy is more than one choice of a leaf $e \in S$ of S to delete. An idea to solve this is to restrict the deletion in reduction (and the addition in generation) to the *maximum* leaf of S. This ensures the reduction sequence $\mathcal{R} = S_0, \ldots, S_\ell$ for S to be unique to each S. We call such a unique sequence the *maximum elimination sequence* for a sub-hypergraph S, and denote by $\mathcal{MES}(S)$.

Lemma 4. *$\mathcal{MES}(S)$ is the unique signature of each connected and Berge-acyclic sub-hypergraph $S \subseteq \mathcal{E}$.*

From this lemma, we can generate S in a unique way by generating $\mathcal{MES}(S)$ instead. Now, we describe our algorithm.

Definition 5. Let S' be any connected and Berge-acyclic subhypergraph S' such that $|S'| \geq 2$. Then, the *parent* of S' is the set $\mathcal{P}(S') = S' - f$, where f is the leaf of S such that $cnt(f, S' - f) = 1$ having the maximum edge ID among all leaves, that is, $f = \max(L(S))$. This condition is called the *maximum leaf condition*. In this case, we call S' a *child* of $S = \mathcal{P}(S')$.

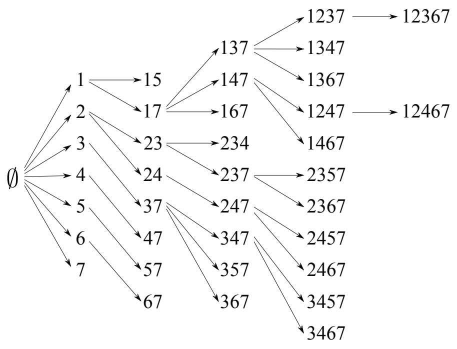


Fig. 3. The family tree for $\mathcal{H}_1$ in Fig. 1

Then, we define our tree-shaped search space. Let $\mathcal{H}$ be an input hypergraph. The *family tree* for the class $\mathcal{CA}$ of connected, and Berge-acyclic sub-hypergraphs of $\mathcal{H}$ is a multi-rooted DAG $\mathcal{T} = (\mathcal{CA}, \mathcal{P}, \mathcal{I})$, where

- $\mathcal{CA}$ is the vertex set of $\mathcal{T}$ that consists of all connected, and Berge-acyclic sub-hypergraphs in an input hypergraph $\mathcal{H}$.
- $\mathcal{P}$ defines the set of reverse edges of $\mathcal{T}$ that assign the parent $\mathcal{P}(S')$ to a child S'.
- $\mathcal{I}$ is the set of all single subsets as the root nodes of $\mathcal{T}$.

The next lemma says that the family tree is actually a tree-shaped search root.

Lemma 5. *For any input hypergraph $\mathcal{H}$, the family tree for $\mathcal{CA}$ on $\mathcal{H}$ is a spanning forest that contains all connected and Berge-acyclic subsets in $\mathcal{CA}$ as its nodes.*

Proof. From Lemma 3, it immediately follows that $\mathcal{P}(S')$ is always connected, and Berge-acyclic, and has size strictly smaller than S'. Since each path of $\mathcal{T}$ is a $\mathcal{MES}$ for some element of $\mathcal{CA}$, $\mathcal{T}$ is connected at some root in $\mathcal{I}$. On the other hand, since each reverse edge strictly reduces the size of S, $\mathcal{T}$ contains no cycle. Hence, the lemma is proved. ■

Example 5. In Fig. 3, we show the family tree for $\mathcal{H}_1$ in Fig. 1. For example, the parent $\mathcal{P}(S_2)$ of S_2 is $\mathcal{P}(P_2) = 137$. Then, there exists the reverse edge from $\mathcal{P}(S_2)$ to its child S_2. $\mathcal{P}(S_2)$ has other children $S_6 = 1237$ and $S_7 = 1367$. Edges $(\mathcal{P}(S_2), S_6)$ and $(\mathcal{P}(S_2), S_7)$ are also the members of the set of reverse edge in the family tree.

In Algorithm 1, we show our basic DFS algorithm BergeMine and its recursive subprocedure BasicRec that finds all connected, and Berge-acyclic sub-hypergraphs in $\mathcal{H}$ in depth-first manner. This algorithm is a simple backtracking algorithm, working as follows. Starting from each singleton subset $\{e\}$ in $\mathcal{I}$,

Algorithm 2. The algorithm for computing the border set of a sub-hypergraph

```
1: procedure ComputeBorder($S$: sub-hypergraph, $\mathcal{V}, \mathcal{E}$)
2:     Output: $Border(S) = \{ f \in (\mathcal{E} \setminus S) \mid cnt(f, S) = 1 \}$.
3:     Mark all vertices of $\mathcal{V}(S)$;
4:     $Border \leftarrow \emptyset$;
5:     for each $e \in \mathcal{E}(S)$ do
6:         Count the number $cnt(e, S)$ of all marked vertices in $e$;
7:         if $cnt(e, S) = 1$ then
8:             $Border \leftarrow Border \cup \{e\}$;
9:     return $Border$;
```

the algorithm searches the family tree $\mathcal{T}$ for connected, and Berge-acyclic subsets by expanding the parent subset S by adding a new leaf f to obtain a child $S' = S \cup \{f\}$. In the expansion, it apply pruning for redundant subsets using the definition of a correct child based on the maximum leaf condition of the child. If expansion is no longer possible, it backtrack to the parent.

To compute the border set, we use the procedure ComputeBorder in Algorithm 2.

Lemma 6. *The algorithm* ComputeBorder *in Fig. 2 computes the border set of an hyperedge subset S $O(||S||) = O(nm)$ time using $O(n)$ additional space.*

We give the time and space complexity of the basic algorithm below.

Theorem 1 (main result). *The algorithm* BergeMine *of Fig. 1 finds all connected Berge-acyclic sub-hypergraphs contained in an input hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ without duplicates in $O(Nm) = O(nm^2)$ delay and $O(N)$ words of space, where $n = |\mathcal{V}|$, $m = |\mathcal{E}|$, and $N = ||\mathcal{E}||$ are the numbers of vertices and hyperedges, and the total size of the hyperedges in $\mathcal{H}$.*

From the theorem, we have the following corollary.

Corollary 1. *The class of all connected Berge-acyclic sub-hypergraphs contained in an input hypergraph $\mathcal{H}$ can be enumerated in polynomial delay and polynomial space in the size of input $\mathcal{H}$.*

4 The Modified Algorithm

In this section, we show a modified version of our depth-first mining algorithm that finds all connected and Berge-acyclic sub-hypergraphs in an input hypergraph $\mathcal{H}$ in $O(|f|\tau(m) + |B|)$ time using $O(N)$ space and preprocessing, where $N = ||\mathcal{E}(\mathcal{H})||$. This algorithm is *adaptive* since its time complexity only depends on the size of the discovered sub-hypergraph S, rather than the whole input. This adaptively is quite important in mining a large hypergraph. In what follows, we use the dynamic data structure of Beame and Fich [5] with operation time $\tau(m) = O(((\log\log m)^2/\log\log\log m))$.

The basic idea of our modified algorithm is an incremental maintenance of the subset *MaxBorder*(S) of hyperedge candidates to insert, called the maximal border hyperedges.

Definition 6. The maximal border of a sub-hypergraph S is the set of hyperedges defined by:

$$\mathit{MaxBorder}(S) = \{\, e \in \mathcal{E} \setminus S \mid \mathit{cnt}(e, S) = 1,\ e = \max L(S \cup \{e\}) \,\}, \qquad (1)$$

that is, *MaxBorder*(S) consists of all hyperedges e of $\mathcal{H}$ satisfying the next conditions: (i) e is a border of S (i.e., $\mathit{cnt}(e, S) = 1$), and (ii) e is the maximum leaf of $S' = S \cup \{e\}$ among all leaves when it is added to S.

In Algorithm 4, we show our modified depth-first algorithm FastBergeMine as well as its recursive subprocedure FastRec for mining all connected, Berge-acyclic sub-hypergraphs in incrementally. By using *MaxBorder*(S), in our depth-first mining algorithm FastBergeMine, we can generate any children $S' = S \cup \{f\}$ from a parent S by just selecting any hyperedge $e \in \mathit{MaxBorder}(S)$ without testing the pruning condition for duplication because the condition is already included by the definition of the maximal border set. In other words, we are eager to make selection of border candidates and test for duplication at the same time in advance.

Therefore, it remains how to efficiently compute the maximal border set. Surprisingly, we can show that this is done in almost optimal time complexity in amortized analysis using a procedure similar to the α-acyclicity test by (Tarjan and Yannakakis [22]). The key to the algorithm is the following recurrence relation for the maximum and the second maximum leaves when we update a parent S by adding a new maximum border $f \in \mathit{MaxBorder}(S)$ to generate a children $S' = S \cup \{f\}$.

Lemma 7. *Let us denote by max*(S) *and 2max*(S) *the maximum and the second maximum leaves of a parent set* $S \subseteq \mathcal{E}$. *Then, the maximum and the second maximum leaves max*(S') *and 2max*(S') *of a child* $S' = S \cup \{f\}$ *satisfy the following recurrence:*

- *If f connects $\ell_{\max} = \mathit{maxLeaf}(S)$:*
  - *If $f >$ 2max(S), then max$(S') \leftarrow f$ and 2max$(S') \leftarrow$ max(S) hold.*
  - *Otherwise, max$(S') \leftarrow$ max(S) and 2max$(S') \leftarrow$ 2max(S) hold.*
- *Otherwise:*
  - *If $f >$ max(S), then max$(S') \leftarrow f$ and 2max$(S') \leftarrow$ max(S) hold.*
  - *Otherwise, max$(S') \leftarrow$ max(S) and 2max$(S') \leftarrow$ 2max(S).*

Proof. In each ease, the proof immediately follows from the case analysis using the definitions of *max*, *2max*, and the maximal border set. ■

From Lemma 7 above, we can update *max*(S) and *2max*(S) incrementally in constant time. Now, we show the algorithm UpdateMaxBorder in Algorithm 3 that incrementally updates the new border est *MaxBorder*$(S \cup \{f\})$ from the older one given the border edge f to add, S, and $\mathit{MB} = \mathit{MaxBorder}(S)$.

Algorithm 3. The algorithm for computing the border set of a sub-hypergraph

```
1: procedure UpdateMaxBorder($f$: hyperedge, $S, B, R$: hyperedge subsets, $\mathcal{H}$: hypergraph )
   Pre-conditions: $S' = S \cup \{f\}$, $f \in R$, $B = MB(S)$, and $R = \mathcal{E}(\mathcal{H}) \setminus S$.
   Output: $MB(S') = \{\, f \in (\mathcal{E} \setminus S') \mid cnt(f, S') = 1, f = \max L(S') \,\}$.
   Global variable: A dynamic data structure $\mathcal{D}$ for storing a hyperedge subset in
   linear space supporting membership, insert, and delete in sublinear time $t = \tau(m)$.
   //Step 1: Update the maximum leaves.
2:     $S' = S \cup \{f\}$;
3:     Update the maximum leaves $max(S')$ and $2max(S')$
         from $f$, $max(S')$, and $2max(S')$ according to Lemma 7.    ▷ in $O(1)$ time
   //Step 2: Update the maximum border set.
4:     $MB(S') \leftarrow \emptyset$;
5:     //Step 2.1: Existing borders other than $f$
6:     for each $e$ in $MB(S) \setminus \{f\}$ do    ▷ $O(|MB(S)|)$ times
7:         Add $e$ to $MB(S')$ if $e = maxL(S' \cup \{e\})$.
8:     //Step 2.2: New borders connecting to $f$
9:     for each vertex $x \in f$ do    ▷ $O(|f|)$ times to $S'$
10:        for each hyperedge id $e \in N(x, \mathcal{D})$ do    ▷ Charge $O(1)$ time to $e$ in $\mathcal{D}$
11:            $cnt[e] \leftarrow cnt[e] + 1$;
12:            if $cnt[e] = 1$ then    ▷ $cnt$ increased from 0 to 1
13:                Add $e$ to the candidate set $\mathcal{D}$;    ▷ Charge $O(\tau(m))$ time to $e$
14:                Add $e$ to $MB(S')$ if $e = maxL(S' \cup \{e\})$.
15:            else if $cnt[e] = 2$ then    ▷ $cnt$ increased from 1 to 2
16:                Remove $e$ from candidate set $\mathcal{D}$;    ▷ Charge $O(\tau(m))$ time to $e$
17:        //Note: each hyperedge is processed at most twice overall
18:    end for
19:    return $MB$;
```

For efficient update, the algorithm uses a dynamic data structure $\mathcal{D}$ for storing a set $\mathcal{D}$ of candidate hyperedges, which is similar to the *DLX* (also known as "*Dancing Links*") data structure by Knuth [15] as described in Sec. 2. Then, we have the next lemma.

Lemma 8. *Let $S \subseteq \mathcal{E}$ be a sub-hypergraph and $f \in R = (\mathcal{E} \setminus S)$ be a maximum border hyperedge of S. Given f, B, and R, the algorithm* UpdateMaxBorder *in Algorithm 3 computes the set MaxBorder$(S \cup \{f\})$ of all maximum border hyperedges of $S' = S \cup \{f\}$ in $O(\Delta MB(S)\tau(m) + |B|)$ time using $O(N)$ space and $O(N)$ preprocessing (at once in the initialization), where $\Delta MB(S)$ is the number of newly added hyperedges to the maximum border of S, $N = ||\mathcal{E}(\mathcal{H})||$, $\tau(m) = ((\log\log m)^2/\log\log\log m)$, r and d are the rank and degree of $\mathcal{H}$, respectively, and $B = MaxBorder(S)$ is the set of all maximum borders of S.*

Proof. Consider Algorithm 3. During the computation of the recursive mining procedure, we maintain the pointers $max(S)$, $2max(S)$, and the dynamic data structure $\mathcal{D}$. From Lemma 7, Step 1 correctly updates the maximum and 2nd maximum leaves in S in constant time. When a new border f is added to S, the

Algorithm 4. The modified algorithm FastBergeMine for mining all connected, Berge-acyclic sub-hypergraphs based on the reverse search

```
 1: procedure FastBergeMine($\mathcal{H} = (\mathcal{V}, \mathcal{E})$: input hypergraph )
 2:     for each hyperedge $e \in \mathcal{E}(\mathcal{H})$ do
 3:         $MB_e \leftarrow \{ f \in \mathcal{E}(\mathcal{H}) \mid (|f \cap e| = 1) \}$;
 4:         $R_e \leftarrow \mathcal{E}(\mathcal{H}) \setminus \{e\}$;
 5:         FastRec($\{e\}, MB_e, R_e, \mathcal{H}$);
 6: procedure FastRec($S, MB, R \subseteq \mathcal{E}(\mathcal{H}), \mathcal{H}$: hypergraph)
 7:     Invariant: $MB = MaxBorder(S)$ and $R = \mathcal{E}(\mathcal{H}) \setminus S$ hold.
 8:     Output $S$;
 9:     for each border hyperedge $f \in MB$ do          ▷ Generation of children
10:         $S' \leftarrow S \cup \{f\}$;
11:         $R' \leftarrow R \setminus \{f\}$;
12:         Incrementally compute $MB' = MaxBorder(S', \mathcal{H})$ from $f$, $MB$, and $R$;
13:         FastRec($S', MB', R', \mathcal{H}$);
14:         Restore the changes on $MB'$;
```

only borders to be changed are (i) f is removed, and (ii) all neighboring hyperedges of f, and (ii) all existing border edges that has a non-empty intersection to f other than its connection point to S. Step 2 handles these cases correctly. For time analysis of Step 2, we observe that during computation from the root hypothesis $\emptyset$ to the current set S, any hyperedge e in $\mathcal{H}$ will be processed at most twice after initialization, that is, it is incremented to $cnt(e) = 1$ at the first time, and it is incremented to $cnt(e) = 2$ the second time. Then, it is removed from $\mathcal{D}$ forever (otherwise a back tracking occurs). We can show that the amortized cost for Step 2 to obtain each the maximum border of S' is at most $O(\Delta MB(S)\tau(m) + |B|)$, where $\tau(m) = ((\log\log m)^2/\log\log\log m)$. ■

From Lemma 8, we show the main theorem of this paper.

Theorem 2 (The adaptive delay by the modified mining algorithm). *The algorithm* FastBergeMine *of Fig. 4 finds all connected Berge-acyclic sub-hypergraphs contained in an input hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ without duplicates in $t = O(\Delta MB(S)\tau(m)) = O(rd \cdot \tau(m))$ amortized delay (amortized time per solution) using $O(N)$ space and $O(N)$ preprocessing, where f is the added edge, B is the maximum border of S, $N = ||\mathcal{E}||$ is the total size of the hyperedges in $\mathcal{H}$, r and d are the rank and degree of $\mathcal{H}$, respectively, and $\tau(m) = ((\log\log m)^2/\log\log\log m)$.*

Proof. The correctness of the algorithm FastBergeMine is obvious from that of the basic algorithm and the definition of *MaxBorder*. From Lemma 8, at each iteration for solution, the computation time of the maximum border is $t = O(\Delta MB(S)\tau(m) + |B|) = O(rd \cdot \tau(m) + |B|)$. By using appropriate charging scheme to each child of S', we can remove the cost $O(|B|)$ since the addition of any maximum border hyperedge to S' always yields a proper solution.

Moreover, we have $\Delta MB(S) = O(|f| \sum_{x \in f} |N(x)|) = O(rd)$. Therefore, the time complexity becomes $t = O(\Delta MB(S)\tau(m)) = O(rd \cdot \tau(m))$ as claimed. ■

From the theorem, we have the following corollary.

Corollary 2. *The class of all connected Berge-acyclic sub-hypergraphs contained in an input hypergraph $\mathcal{H}$ can be enumerated in amortized delay that depends only on the number of newly added maximum border hyperedges of a discovered subset S using polynomial space and preprocessing in the input size $||\mathcal{E}(\mathcal{H})||$.*

It is an open question whether there exists some enumeration algorithm whose amortized delay depends only on the number of difference of a discovered subset.

5 Conclusion

In this paper, we considered the problem of finding all all connected Berge-acyclic sub-hypergraphs contained in an input hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ without duplicate with applications to generalization of itemset mining from transaction databases and also discovering connected substructures from datasets in the form of sets of sets. As main results, we presented an efficient DFS algorithm for the problem that achieves polynomial delay and space complexity. We also presented an improved algorithm that has adaptive delay depending only on the size of discovered sub-hypergraph.

In this paper, we focused on only the theoretical aspect of the problem. One of the most important future researches is implementation and empirical evaluation of the proposed algorithms on artificial and real datasets. It is also an important problem to find suitable application of this problem in knowledge discovery problems in the real world including knowledge discovery from mobility data or social networks. We want to study these problems in future.

Acknowledgements. The authors would like thank anonymous reviewers for their comments which improved the correctness and the presentation of this paper very much, and thank Shin-ichi Nakano, Kunihiko Sadakane, and Tetsuji Kuboyama for helpful discussions and comments. This research was partly supported by MEXT Grant-in-Aid for Scientific Research (A), 24240021, FY20122015, and Grant-in-Aid for JSPS Fellows (25·1149).

References

1. Arimura, H., Uno, T.: An efficient polynomial space and polynomial delay algorithm for enumeration of maximal motifs in a sequence. Journal of Combinatorial Optimization 13, 243–262 (2006)
2. Arimura, H., Uno, T.: Mining maximal flexible patterns in a sequence. In: Satoh, K., Inokuchi, A., Nagao, K., Kawamura, T. (eds.) JSAI 2007. LNCS (LNAI), vol. 4914, pp. 307–317. Springer, Heidelberg (2008)

3. Arimura, H., Uno, T.: Polynomial-delay and polynomial-space algorithms for mining closed sequences, graphs, and pictures in accessible set systems. In: Proceedings of the SIAM Int'l Conf. on Data Mining 2009 (SDM 2009), pp. 1087–1098 (2009)
4. Avis, D., Fukuda, K.: Reverse search for enumeration. Discrete Applied Math. 65, 21–46 (1993)
5. Beame, P., Fich, F.E.: Optimal bounds for the predecessor problem and related problems. Journal of Computer and System Sciences 65(1), 38–72 (2002)
6. Berge, C., Minieka, E.: Graphs and hypergraphs, vol. 7. North-Holland (1973)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. The MIT Press (2001)
8. Daigo, T., Hirata, K.: On Generating All Maximal Acyclic Subhypergraphs with Polynomial Delay. In: Nielsen, M., Kučera, A., Miltersen, P.B., Palamidessi, C., Tůma, P., Valencia, F. (eds.) SOFSEM 2009. LNCS, vol. 5404, pp. 181–192. Springer, Heidelberg (2009)
9. Fagin, R.: Degrees of acyclicity for hypergraphs and relational database schemes. Journal of the ACM 30(3), 514–550 (1983)
10. Ferreira, R., Grossi, R., Marino, A., Pisanti, N.: Optimal Listing of Cycles and st-Paths in Undirected Graphs (2012)
11. Ferreira, R., Grossi, R., Rizzi, R.: Output-sensitive listing of bounded-size trees in undirected graphs. In: Demetrescu, C., Halldórsson, M.M. (eds.) ESA 2011. LNCS, vol. 6942, pp. 275–286. Springer, Heidelberg (2011)
12. Hirata, K., Kuwabara, M., Harao, M.: On finding acyclic subhypergraphs. In: Liśkiewicz, M., Reischuk, R. (eds.) FCT 2005. LNCS, vol. 3623, pp. 491–503. Springer, Heidelberg (2005)
13. 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)
14. Kawasoe, S., Sakamoto, H., Arimura, H., Arikawa, S.: Efficient substructure discovery from large semi-structured data. IEICE Transactions on Information and Systems 87(12), 2754–2763 (2004)
15. Knuth, D.E.: Dancing links. eprint arXiv:cs/0011047 (November 2000)
16. Kuboyama, T., Hirata, K., Aoki-Kinoshita, K.F.: An efficient unordered tree kernel and its application to glycan classification. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 184–195. Springer, Heidelberg (2008)
17. Li, X.-L., Tan, S.-H., Foo, C.-S., Ng, S.-K., et al.: Interaction graph mining for protein complexes using local clique merging. Genome Informatics 16(2), 260 (2005)
18. Lovász, L.: Matroid matching and some applications. Journal of Combinatorial Theory, Series B 236, 208–236 (1980)
19. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.-C.: Mining sequential patterns by pattern-growth: The prefixspan approach. IEEE TKDE 16(11), 1424–1440 (2004)
20. Shioura, A., Tamura, A., Uno, T.: An optimal algorithm for scanning all spanning trees of undirected graphs. SIAM J. Comput. 26(3), 678–692 (1997)
21. Silva, A., Meira Jr., W., Zaki, M.J.: Structural correlation pattern mining for large graphs. In: Proceedings of the Eighth Workshop on Mining and Learning with Graphs, pp. 119–126. ACM (2010)

22. Tarjan, R., Yannakakis, M.: Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs. SIAM Journal on Computing 13(3) (1984)
23. Tarjan, R.E., Read, R.C.: Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. Networks 5(3), 237–252 (1975)
24. Uno, T., Arimura, H.: Ambiguous frequent itemset mining and polynomial delay enumeration. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 357–368. Springer, Heidelberg (2008)
25. Uno, T., Asai, T., Uchida, Y., Arimura, H.: An efficient algorithm for enumerating closed patterns in transaction databases. In: Suzuki, E., Arikawa, S. (eds.) DS 2004. LNCS (LNAI), vol. 3245, pp. 16–31. Springer, Heidelberg (2004)
26. Wasa, K., Kaneta, Y., Uno, T., Arimura, H.: Constant time enumeration of bounded-size subtrees in trees and its application. In: Gudmundsson, J., Mestre, J., Viglas, T. (eds.) COCOON 2012. LNCS, vol. 7434, pp. 347–359. Springer, Heidelberg (2012)
27. Yan, X., Han, J.: gspan: Graph-based substructure pattern mining. In: Proceedings of the 2002 IEEE International Conference on Data Mining, ICDM 2003, pp. 721–724. IEEE (2002)
28. Zaki, M.J.: Scalable algorithms for association mining. IEEE Transactions on Knowledge and Data Engineering 12(3), 372–390 (2000)
29. Zaki, M.J., Hsiao, C.-J.: Efficient algorithms for mining closed itemsets and their lattice structure. IEEE Transactions on Knowledge and Data Engineering 17(4), 462–478 (2005)

Hyperlink Prediction in Hypernetworks Using Latent Social Features*

Ye Xu[1], Dan Rockmore[1,2,3], and Adam M. Kleinbaum[4]

[1] Computer Science Department, Dartmouth College
[2] Department of Maths, Dartmouth College
[3] The Santa Fe Institute
[4] Tuck School of Business, Dartmouth College
{ye,rockmore}@cs.dartmouth.edu,
adam.m.kleinbaum@tuck.dartmouth.edu

Abstract. Predicting the existence of links between pairwise objects in networks is a key problem in the study of social networks. However, relationships among objects are often more complex than simple pairwise relations. By restricting attention to dyads, it is possible that information valuable for many learning tasks can be lost. The *hypernetwork* relaxes the assumption that only two nodes can participate in a link, permitting instead an arbitrary number of nodes to participate in so-called *hyperlinks* or *hyperedges*, which is a more natural representation for complex, multi-party relations. *However, the hyperlink prediction problem has yet to be studied.* In this paper, we propose HPLSF (Hyperlink Prediction using Latent Social Features), a hyperlink prediction algorithm for hypernetworks. By exploiting the homophily property of social networks, HPLSF explores social features for hyperlink prediction. To handle the problem that social features are not always observable, a latent social feature learning scheme is developed. To cope with the arbitrary cardinality hyperlink issue in hypernetworks, we design a feature-embedding scheme to map the a priori arbitrarily-sized feature set associated with each hyperlink into a uniformly-sized auxiliary space. To address the fact that observed features and latent features may be not independent, we generalize a structural SVM to learn using both observed features and latent features. In experiments, we evaluate the proposed HPLSF framework on three large-scale hypernetwork datasets. Our results on the three diverse datasets demonstrate the effectiveness of the HPLSF algorithm. Although developed in the context of social networks, HPLSF is a general methodology and applies to arbitrary hypernetworks.

Keywords: Hypernetworks, Hyperlink Prediction, Social Features.

* Dan Rockmore was partially supported by AFOSR Award FA9550-11-1-0166 and the Neukom Institute for Computational Science at Dartmouth College. Ye Xu was partially supported by a grant from the Alfred P. Sloan Foundation.

1 Introduction

Networks provide a powerful framework for modeling real world relationships in which vertices represent objects and links between pairs of vertices indicate their interaction [29]. Nevertheless, in many real world problems, the natural relationships encoding the phenomenon may exist among more than two objects or actors. Examples include buyer-broker-seller triads in a market relationship [3], or subsets of co-expressed genes in a genetic network [16]. In such cases limiting the relationships to dyads may obscure valuable information for learning tasks. The *hypernetwork* is a combinatorial structure in which *hyperlinks* or *hyperedges* represent a relationship that can exist among more than three objects (see e.g., [42]) and thus can provide representation for complex relationships (a hyperlink relating only two actors would simply be a link in the usual network sense). Due to its powerful modeling ability, the hypernetwork framework has attracted attention in a variety of application domains, including scene classification[34], bioinformatics[16], finance[3], and sociology[5].

Link prediction techniques [21,22,8,1] aim to predict the existence of links between vertices in a network. It is an important task in many areas, especially social networks. Thus, it is then natural and as useful to consider the analogous *hyperlink prediction problem* in the hypernetwork setting. A significant difference and challenge in the hypernetwork setting is the a priori arbitrary cardinality of each hyperlink (i.e., the number of nodes associated with the hyperlink). To the best of our knowledge, hyperlink prediction remains untouched in the hypernetwork scenario.

In this paper we address the hyperlink prediction problem in the context of social networks. Social networks often exhibit *homophily* [25], wherein people with similar social affiliations or properties show a preference for interacting with each other. For example, in a college, connections are more likely to exist among students who co-enroll in a class or join in the same sports team or group. A few works [12,27] indicate that considering these social affiliations or features can improve the accuracy for link prediction tasks. Unfortunately, these social affiliations or features are not always observable. By ignoring the "latent" social features (as is done in a few current link prediction algorithms [37,22]), it is possible to lose important information for link predictions. Therefore, it is desirable to utilize these latent social features in link prediction methods. However, finding a means of exploring the "latent" social features is a thorny issue and there is limited research on this in the link prediction literature, let alone hyperlink prediction.

In this paper, we propose HPLSF (Hyperlink Prediction using Latent Social Features), a link prediction algorithm for hypernetworks. Although developed in the context of social networks, HPLSF is a general methodology is readily generalized to arbitrary hypernetworks. Following the homophily property of social networks, we utilize social features for hyperlink prediction. To cope with the problem that social features are often unobservable, we design a scheme to learn latent social features for each individual vertex, each dimension of which is indicative of a plausible social affiliation for the vertex. This transforms the hyperlink prediction problem into a classification task, where latent social features and observed features (if available) are utilized together for hyperlink prediction. The fact that hyperlinks can have arbitrary size (given by the number of actors connected in the hyperlink) raises an additional challenge. We attack this by designing a feature embedding method to map the feature set of the nodes associated with each potential hyperlink into an auxiliary space. In this case, uniformly-sized feature sets are learned from the a priori arbitrarily-sized feature sets. In the last step a structural SVM classifier is generalized under the observed features and latent features after feature embedding because interdependent relationships may exist between observed features and latent features.

In summary, the contributions of this paper are as follows: (1) We design an algorithm to predict the existence of hyperlinks in hypernetworks. As far as we know, HPLSF is the first hyperlink prediction work for hypernetworks. HPLSF can be generalized into any type of hyperneworks although in this paper, we employ email hypernetworks for evaluation. (2)We develop a scheme to learn latent social features for each individual vertex in the hypernetwork. In this way, the *homophily* property of social networks can be fully utilized when considering hyperlink prediction for hypernetworks. (3) We propose a novel feature-embedding strategy to cope with the arbitrarily-sized hyperlink cardinality challenge. Contrary to traditional link prediction work [22], we do not consider the feature set extracted from the group of nodes associated with one potential link/hyperlink directly. Instead, we design a scheme to map this feature set into an embedding space, and each dimension of the embedding space reflects the interaction strength of the group of nodes. In this case, the arbitrarily-sized feature extracted from each hyperlink is mapped into a uniformly-sized feature. (4) We propose to employ structural SVM to learn with both observed features and latent features after feature-embedding in case that observed features and latent features are not necessarily independent. (5) We deploy these ideas on three large-scale email hypernetwork datasets from diverse sources: a large university, an urban-centered hospital, and a large IT corporation. It is the first time that these three datasets are considered in the hypernetwork setting. The heterogeneity of these contexts validate the effectiveness of the proposed HPLSF.

The rest of the paper is organized as follows. In Section 2, we briefly introduce related work. The detailed HPLSF framework is proposed in Section 3. In Section 4 we report experimental results. Finally in Section 5, we conclude the paper.

2 Related Work

Hypernetworks (see e.g.,[42]) have drawn significant attention in various domains. For instance, in [10] hypernetworks are used to model DNA interactions wherein they achieve better disease detection accuracy as compared with using traditional networks. In [3], the hypernetwork is employed to model the correlations of daily stock prices, thereby improving the stock price prediction accuracy. Sun et al. [34] model the set of multiple labels along with the labels' correlations under the multi-instance setting via hypernetworks. Because the *high-order relations* in multi-labels [15] can be captured by hyperlinks, the classification performance is competitive. However, all existing works in the current hypernetwork literature assume constant cardinality for the hyperlinks over the whole hypernetwork. Additionally, these previous researches focus on utilizing the hyperlink relationships to infer the labels of individual nodes in hypernetworks, rather than doing hyperlink predictions in hypernetworks. In our paper, we consider the problem of hyperlink prediction for hypernetworks, and allow for arbitrary (and varying) cardinality of the hyperlink over the hypernetwork.

Our work also relates to social feature learning. Hopcroft et al. [13] indicate that social features reflect the *homophily* property and play an important role in social networks. However, social features are not always observable. There have been various efforts exploring methods to learn the "latent" social features. Neville and Jensen [28] utilize a clustering scheme to achieve a membership vector of each person in the network. In [35], a set of social features is learned using a graph cutting method to help classify relational data in networks. It is worth noting that these works all take advantage of social features to improve the classification performance under the *relational learning* setting [23], i.e., classifying each individual datum (vertex) in a network where data are no longer assumed to be independently and identically distributed. Our work however, aims to utilize the latent social features to predict the existence of hyperlinks in hypernetworks.

Recently, there are a few link prediction models proposed for traditional networks that use "latent" social features [12,11,27,43]. These latent feature models assume that each object (vertex) in the network belongs to a set of latent classes. Thus, the latent class membership of each individual object is useful for predicting pairwise links between objects in networks. Note that these works model latent features according to pairwise relations between vertices in the network. Statistical methods such as variational inference or sampling are used in training and inference. These are time-consuming and prone to suffering from the *local maxima problem*. By contrast, our paper explores the "latent" social features for vertices based on the distance information conveyed in hypernetworks, and employs a simple clustering technique.

Feature embedding [19], which maps a fixed set data into a feature space, is a powerful tool in machine learning. Previous feature embedding techniques were designed for a few particular learning tasks. For example, Kondor [19] developed a feature embedding algorithm for image classification. Grangier [9] employed feature embedding to deal with incomplete data in the original dataset. In our work, we design a feature embedding method to address the arbitrary-sized hyperlink cardinality issue in hypernetwork. As far as we known, it is the first time feature embedding techniques have been used in the network/hypernetwork scenario.

Another line of related work is community detection [30]. Community detection focuses on dividing the vertices in a network into several groups by only using the information encoded in the network topology. It is a hot topic in network study and a few methods have been proposed [31,30]. However, there are fundamental differences between community detection and link/hyperlink prediction algorithms. Almost all community detection algorithms only consider topological information from networks, while our proposed hyperlink prediction method aims to utilize both observed and latent social features from nodes (objects) within the network.

3 Hyperlink Prediction Using Latent Social Features

In what follows, we give the description of HPLSF framework, which takes advantage of the observed information as well as the latent social features from each individual vertex in the hypernetwork.

3.1 Hyperlink Prediction Problem

Before presenting the hyperlink prediction problem in detail, we give the formal description of hypernetworks as follows. A hypernetwork is formalized as an ordered pair $H = (V, E)$, where $V = \{\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_n}\}$ is the set of vertices, and $E = \{\boldsymbol{e_1}, \boldsymbol{e_2}, ..., \boldsymbol{e_m}\}$ is the set of hyperlinks (hyperedges). Therein, if $\boldsymbol{e_i} = \{\boldsymbol{v_{i_1}}, \boldsymbol{v_{i_2}}, ..., \boldsymbol{v_{i_k}}\}$ is a hyperlink with $k > 2$, it is then different from a link (edge) in the traditional network setting because the number of associated vertices could be more than 2. An example of hypernetwork [42] is given in Fig.1.

If all the hyperlinks in the hypernetwork H have the same cardinality k, then the H is a k-uniform hypernetwork, otherwise, H is an *arbitrary-sized hypernetwork*. Although most hypernetwork applications [10,34] can only handle k-uniform hypernetworks, in this paper, we propose a hyperlink prediction framework on arbitrary-sized hypernetworks. The task of link prediction for hypernetworks can thus be formulated as follows: Given a training dataset $S = \{(\boldsymbol{e_1}, y_1), (\boldsymbol{e_2}, y_2), ..., (\boldsymbol{e_t}, y_t)\}$, where $\boldsymbol{e_i}$ represents a possible relation among several vertices, and $y_i \in \{-1, +1\}$ represents the label of the $\boldsymbol{e_i}$ (i.e., if $y_i = +1$, there exists a hyperlink among the set of vertices; if $y_i = -1$, there is no hyperlink.), the goal is to learn the labels in the test set $T = \{\boldsymbol{e_{t+1}}, \boldsymbol{e_{t+2}}, ..., \boldsymbol{e_{t+u}}\}$.

3.2 Exploring Latent Social Features

As we have mentioned, homophily (the idea that people with similar attributes are more likely to interact with each other) is an important characteristic in social networks. Several relational learning works (see e.g., [35]) show that utilizing the homophily property of social networks in the course of learning social features can improve the classification accuracy for network data. A few researchers [12,27] suggest that social features also play a significant role in predicting pairwise links for traditional networks. In hypernetwork scenarios, it is then also natural to consider the homophily property, and thus to take advantage of social features for hyperlink prediction.

Unfortunately, social features are often unobservable. Thus it is not trivial to obtain “latent” social features. In hypernetworks, each social feature indicates (to some extent) a particular property or affiliation for objects. Note that objects

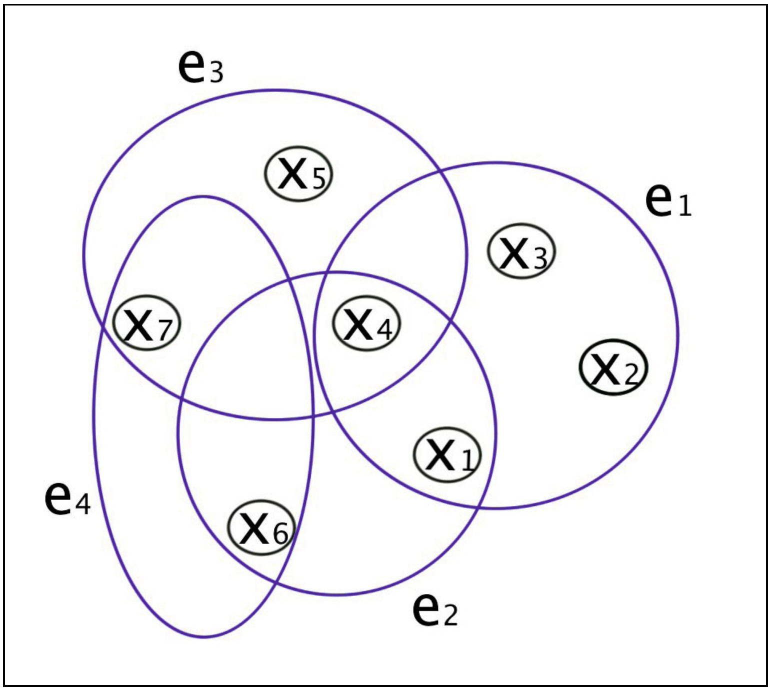


Fig. 1. An example of hypernerwork. The hypernetwork H consists of seven vertices and five hyperlinks. Specifically, $H = (V, E)$, where $V = \{\boldsymbol{x_1}, \boldsymbol{x_2}, ...\boldsymbol{x_7}\}$ and $E = \{\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3}, \boldsymbol{e_4}\}$. Each of the hyperlink could be associated with more than 2 vertices, namely, $\boldsymbol{e_1} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_4}\}$, $\boldsymbol{e_2} = \{\boldsymbol{x_1}, \boldsymbol{x_4}, \boldsymbol{x_6}\}$, $\boldsymbol{e_3} = \{\boldsymbol{x_4}, \boldsymbol{x_5}, \boldsymbol{x_7}\}$, and $\boldsymbol{e_5} = \{\boldsymbol{x_6}, \boldsymbol{x_7}\}$. Because the number of vertices associated with $\boldsymbol{e_1}$ is 4, we say that the cardinality of hyperlink $\boldsymbol{e_1}$ is equal to 4.

(vertices) sharing similar properties or affiliations in hypernetworks interact at a higher rate than dissimilar objects, and are likely to form groups with more frequent within-group interactions. This is naturally associated with *graph partition* [4], a basic task in *graph theory* [32] which focuses on clustering vertices into groups such that within-group interactions are more frequent than between-group interactions. In this way, the description of affiliation in each particular group is considered as one dimension of social features. Many algorithms [4,33] have been investigated for graph partition, among which clustering-based methods play an important role. Based on the intuition that objects that are similar in group affiliations are very likely to be close in a geometric representation, these clustering-based methods construct a geometric embedding to indicate group affiliations for objects. Therefore, in our work, we employ multidimensional scaling (MDS) [6], a typical geometric embedding learner to explore latent social features based on hypernetwork distance.

Multidimensional scaling (MDS) constructs a geometric embedding (feature vector) preserving as best as possible the original distance among data (objects). Each dimension of the MDS embedding indicates the strength of a certain group affiliation [6] and can be regarded as one social feature. Specifically, for a hypernetwork with N vertices, MDS finds the embedding matrix $\mathbf{Z} \in \mathcal{R}^{N\times p}$ (p is the dimensionality of latent features) whose row vectors are group affiliation descriptions (and thus are treated as latent social features) for the corresponding object in the network as follows,

$$\arg\min_{\mathbf{Z}} \|D - \mathbf{Z}\mathbf{Z}^T\|_F. \tag{1}$$

Therein, $\|\cdot\|_F$ denotes the Frobenius norm, and D is the distance matrix obtained from the hypernetwork (D_{ij} is the length of the shortest path from vertex i to j in the hypernetwork H).

As per [6] we can solve (1) as follows: Let Σ be the matrix of the eigenvectors of D , and Λ be a diagonal matrix with the corresponding eigenvalues. The matrix of the p top eigenvalues is denoted by Λ_p and the corresponding columns of Σ is denoted by Σ_p. Then we can obtain the solution of (1) by,

$$\mathbf{Z} = \Sigma_p \Lambda_p^2 \tag{2}$$

It deserves mentioning that when computing D, we set a shortest path maximum length of five hops in order to avoid the full N^2 computation of all-shortest paths. [6] indicates that this approximation scheme can achieve close result compared with full computation.

3.3 Embedding Features into Uniform-Sized Space

Most existing hypernetwork applications [42,10,34], if not all, simply assume that the cardinality of all hyperlinks in hypernetworks is uniform. Obviously this assumption does not hold under many scenarios and thus limits the application scopes of hypernetworks in real-world. In our work, to address this arbitrary-sized hyperlink cardinality challenge, we propose a feature embedding technique to map feature set from all nodes associated with one potential hyperlink into an embedding space. The dimension of the embedding space is uniform and independent of the cardinality of each particular hyperlink. Therefore, it is convenient to train classifiers under the embedded features for hyperlinks with various cardinality.

Our paper focuses on hyperlink predictions by leveraging social features, where each dimension represents one particular social affiliation for the person(node) in the hypernetwork. We aim to learn a feature embedding from the original social feature set of all persons associated with a potential hyperlink, and the mapped feature in the embedding space is supposed to reflect the discriminability of interaction strength among the group of persons. In this work, we employ *entropy impurity* to measure the similarity strength among a group of people based on the values of each dimension of their social features. If all the group people share similar/close value over one particular social feature, it indicates that those persons are similar over the particular social characteristic and the entropy score would be small. On the contrary, if the values of the particular social feature are diverse, the similarity strength of these people are weak and the entropy score would be large.

In particular, given the social feature set $\{\boldsymbol{z_1}, \boldsymbol{z_2}, ..., \boldsymbol{z_k}\}$ (i.e., features calculated by Eq.(2)) from all nodes associated with one potential hyperlink ($\boldsymbol{z_i} \in \mathcal{R}^{\mathcal{M}}$), we construct a map $f : \mathcal{R}^{M \cdot k} \to \mathcal{R}^M$ as follows,

$$f(\boldsymbol{z_1}, \boldsymbol{z_2}, ..., \boldsymbol{z_k}) = [-\sum_{j=1}^{k} p(z_{j1}) logp(z_{j1}), ..., -\sum_{j=1}^{k} p(z_{jM}) logp(z_{jM})] \quad (3)$$

Here, $p(z_{ji})$ is the fraction of feature values at the i^{th} social feature that belong to category z_{ji}, and $-\sum_{j=1}^{k} p(z_{ji}) logp(z_{ji})$ is the entropy score over the total k associated people in the potential hyperlink for the i^{th} social feature.

The designed feature embedding scheme offers great flexibility. First, it can accommodate the hypernetworks with arbitrary-sized hyperlink cardinality. After feature embedding, potential hyperlink features in different dimensionality can be mapped into uniform-sized features (in $\mathcal{R}^M$). Second, the embedding technique accommodates both discrete and continuous social features. When the original feature set contains a mix of discrete and continuous values, the entropy score computing scheme naturally handles both of the two types of values. [2] (For continuous values, we can use a threshold based method when calculating the entropy.)

3.4 Learning with Observed and Latent Features

HPLSF aims to conduct hyperlink prediction by leveraging not only observed features but also latent features. In this subsection, we introduce the details of learning with observed and latent features.

After obtaining the embedded latent features (via the method discussed in last subsection) and the observed features[1], we predict whether a hyperlink exists among a set of vertices. The observed features and latent features are not necessarily independent of each other. Therefore, simply combining the observed-features-based classifier and latent-features-based classifier ignores the potential dependence between the output spaces of the two classifiers and might lead to inaccurate prediction results. Different from classic SVM addressing independent output applications [39,40,38], the structural SVM [36] was designed for learning problems involving dependent outputs. Therefore, in our work, a structural SVM based classifier is generalized to capture the potential interdependent relationship between the outputs of the two classifiers.

[1] We assume that the observed features – metadata – are available for vertices in the network.

Structural SVM employs margins between the true structure $\boldsymbol{y}^\star$ and other possible structures $\boldsymbol{y}$:

$$\forall \boldsymbol{y} \in \mathcal{Y}: \ \boldsymbol{w}^\top \phi(\boldsymbol{x}, \boldsymbol{y}^\star) \geq \boldsymbol{w}^\top \phi(\boldsymbol{x}, \boldsymbol{y}) + \Delta(\boldsymbol{y}^\star, \boldsymbol{y}) - \xi \tag{4}$$

Therein, $\xi > 0$ is a slack variable that controls the tradeoff between satisfying the constraints and optimizing the objectives. $\phi(\boldsymbol{x}, \boldsymbol{y})$ is a joint feature map that characterizes the relation between an input $\boldsymbol{x}$ and an output structure $\boldsymbol{y}$. The loss function $\Delta(\boldsymbol{y}^\star, \boldsymbol{y})$ quantifies the loss associated with the prediction $\boldsymbol{y}^\star$ when the true output is $\boldsymbol{y}$. In structural SVM, two different structures $(\boldsymbol{y}, \boldsymbol{y}^\star)$ could exhibit similar accuracy, which is reflected in the margin constraint. The violation of margin constraints with high loss $\Delta(\boldsymbol{y}^\star, \boldsymbol{y})$ should be penalized more severely than the violation involving the output value with smaller loss.

In our hypernetwork scenario, we denote $\boldsymbol{e^o}$ as the embedded observed features of the potential hyperlink $\boldsymbol{e}$ (i.e., set of vertices), and $\boldsymbol{e^l}$ as the embedded latent features of the potential hyperlink $\boldsymbol{e}$. Here, $\boldsymbol{e^o} = f(\boldsymbol{x_1^o}, \boldsymbol{x_2^o}, ..., \boldsymbol{x_k^o})$ is the embedding of observed features from each individual vertex belonging to the vertices set $\boldsymbol{e}$, and $\boldsymbol{e^l} = f(\boldsymbol{x_1^l}, \boldsymbol{x_2^l}, ..., \boldsymbol{x_k^l})$ is the embedding of latent features from each individual vertex. Meanwhile, we define $\boldsymbol{y} = [y^o, y^l]$ where y^o is the output corresponding to embedded observed features $\boldsymbol{e^o}$ and y^l is the output corresponding to embedded latent features $\boldsymbol{e^l}$.[2] We define the loss function of form

$$\Delta(\boldsymbol{y}^\star, \boldsymbol{y}) = \frac{1}{2}(\mathbf{1}(y^{\star o} \neq y^o) + \mathbf{1}(y^{\star l} \neq y^l)) \tag{5}$$

where $\mathbf{1}(S) = 1$ if S is true; otherwise, $\mathbf{1}(S) = 0$. The defined $\Delta(\boldsymbol{y}^\star, \boldsymbol{y})$ is non-negative and bounded in $[0, 1]$. This loss function supports flexible notions of structural correctness and has been widely used in many structural SVM work [14,24].

Thus the joint feature map for the structural SVM can be written as follows:

$$\Phi(\boldsymbol{e}, \boldsymbol{y}) = [\phi_o(\boldsymbol{e^o}, y^o),\ \phi_l(\boldsymbol{e^l}, y^l)] \tag{6}$$

where $\phi_o(\boldsymbol{e^o}, y^o)$ is the feature map describing the relation between observed features of a potential hyperlink and its corresponding output, and $\phi_l(\boldsymbol{e^l}, y^l)$ is the feature map describing the relation between latent features and its corresponding output.

Using the joint feature map defined in Eq.(6), the constraints for the HPLSF can be formulated as the similar form as Eq.(4). By adding the objective function which minimizes the combination of the regularization term and the penalty term for slack variables, the optimization problem can be written as follows:

$$\min_{\boldsymbol{w}, \boldsymbol{\xi}} \tfrac{1}{2}\|\boldsymbol{w}\|^2 + \tfrac{C}{N}\textstyle\sum_{i=1}^N \xi_i \tag{7}$$

$$\forall i, \forall \boldsymbol{y} \in \mathcal{Y}\backslash \boldsymbol{y_i} : \boldsymbol{w}^\top \Phi(\boldsymbol{e_i}, \boldsymbol{y}) \geq \boldsymbol{w}^\top \Phi(\boldsymbol{e_i}, \boldsymbol{y_i}) + \Delta'(\boldsymbol{y}, \boldsymbol{y_i}) - \xi_i \tag{8}$$

[2] In our work, for each $\boldsymbol{e_i}$, we use 1-Nearest Neighbor Classifier (1-NNC) as the observed-features-based classifier and the latent-features-based classifier to compute y_i^o and y_i^l respectively.

where C is the parameter controlling the tradeoff between satisfying the constraints and minimizing the regularization term. In all our experiments, we set up $C = 1$. The optimization method discussed in [14] is employed to solve the problem (8).

3.5 Summary of the Proposed Algorithm

```
1: **Input:** Hypernetwork $H = (V, E)$, where $V$ is the vertex set and $E$ is the set
   of hyperlinks.
2: Construct distance matrix $D$ from $H$, where $D_{ij}$ indicates the length of the
   shortest path from vertex $i$ to $j$ in the hypernetwork.
3: Calculate latent features for each object in the hypernetwork using Eq.(2).
4: Embed the observed feature set and the latent feature set for each hyperlink
   respectively using Eq.(3).
5: Construct the joint feature maps as Eq.(6) using embedded latent features
   and embedded observed features.
6: Solve the problem (8) via the SVM$^{struct}$.
```

Algorithm 1. Hyperlink Prediction Using Latent Social Features

In this section, we present the detailed HPLSF in Algorithm 1. First, we construct the distance matrix D whose element D_{ij} gives the length of the shortest path between two vertices in the hypernetwork. Then we apply MDS algorithm to calculate the embedding matrix, where each row vector represents the latent feature for the corresponding vertex (object). After obtaining the embedded latent features, we can use them together with the embedded observed features to construct the constraints of the optimization problem (8). Lastly, SVM^{struct} [14] is employed to train the classifier. Note that the proposed prediction framework can be generalized into any other type of hypernetworks although in the experiments we only evaluate it using email hypernetworks.

4 Experiments

In what follows, we introduce three real-world hypernetwork datasets to evaluate the proposed HPLSF framework. The first dataset was collected in a major urban hospital over one year [7]. The second dataset was collected in a large university over 6 semesters (three years) [20,41]. The last dataset was collected from a large IT corporation over three years [18,17]. Because there is no other hyperlink prediction algorithm for hypernetworks with which we can compare our results, we compare HPLSF with three baseline methods. Ob-Model (the classifier trained using embedded observed features) is used as the baseline to demonstrate the importance and necessity of exploring latent social features for hyperlink prediction. We also execute Ex-Model-MDS (the classifier trained using embedded latent features that are learned by the same MDS procedure as HPLSF) and Ex-Model-LFRM

Table 1. The overview of the hospital collaborative hypernetwork datasets

Hyperlink Cardinality	#Hyperlinks
3	735
4	510
5	341
6	245
Arbitrary-Sized	1831

Table 2. The hyperlink prediction accuracy (%) using HPLSF and Ex-Model-LFRM under the hospital collaborative hypernetwork datasets

Hyperlink Cardinality	HPLSF	Ex-Model-LFRM
3	$\mathbf{90.6 \pm 0.4}$	86.2 ± 1.1
4	$\mathbf{88.9 \pm 0.9}$	81.5 ± 2.3
5	$\mathbf{84.3 \pm 2.2}$	74.3 ± 3.5
6	$\mathbf{85.6 \pm 0.4}$	73.7 ± 0.4
Arbitrary-Sized	$\mathbf{80.6 \pm 1.3}$	71.2 ± 1.8

(the classifier trained using embedded latent features that are learned by the Latent Feature Relational Model (LFRM) proposed in [27]) in order to validate the effectiveness of the designed latent social feature learning scheme in HPLSF. Note that LFRM was designed to model latent features under the traditional "two-vertex-link" setting. Thus in our experiments, we transform the hypernetwork into the traditional "two-vertex-link" network (linking all pairs of vertices contained in a particular hyperlink) when using LFRM.

4.1 Hospital Message Hypernetwork Dataset

In this subsection, we consider HPLSF in the context of an email dataset derived from communications in an urban hospital [7]. The hypernetwork contains message communications among patients, their family members, clinicians, and researchers who work on coming up with a cure for particular diseases in the hospital. The messages are collected among 11,944 people over one year via an internal message system in the hospital. The people involved in the message system are treated as the vertex set and all persons that appear in one message are regarded as the set of vertices of a hyperlink. Most hyperlinks in the dataset has a cardinality no larger than 6. In this experiment, we respectively consider $3-$cardinality, $4-$cardinality, $5-$cardinality, $6-$cardinality, and arbitrary-sized cardinality when constructing the hypernetwork. In other word, in the $k-$cardinality hypernetwork, the messages containing exactly k people are considered and others are discarded, while in the arbitrary-sized cardinality hypernetwork, we consider all hyperlinks with any cardinality value. Meanwhile, we randomly sample sets of nodes to form negative hyperlinks. The detailed information about this dataset can be found in Table 1.

Due to the strict privacy regulations in the hospital, the content of each message is discarded. Additionally, any personal information for each person (vertex) in the message system can not be accessed. Therefore, in this dataset, HPLSF only applies latent features to predict hyperlinks. (In this case, HPLSF is equivalent to Ex-Model-MDS.) To demonstrate the effectiveness of the latent feature learning of HPLSF, we execute Ex-Model-LFRM under the hospital collaborative hypernetwork dataset. In this experiment, the latent feature dimension p is determined by this: we use five times 10-fold cross validation to tune this parameter for Ex-Model-LFRM. Then for HPLSF, we use the same value of p as in Ex-Model-LFRM. After obtaining the latent features, SVM^{struct} [14] is employed for training and testing. The 10-fold cross validation scheme is used to achieve the average prediction accuracy, which is listed in Table 2.

The results in Table 2 indicates that HPLSF is able to obtain higher prediction accuracy than Ex-Model-LFRM under all the conditions. Note that as the hyperlink cardinality k increases, the difference between the accuracy of two methods grows. This fact implies that the proposed latent feature learning scheme in HPLSF outperforms LFRM [27] under the hyperlink prediction scenario. It is because that LFRM was designed for the traditional networks and may fail in modeling hyperlink relations. As for the arbitrary-sized hypernetwork, the difference between our HPLSF and the baseline is also significant. Pairwise t-tests at 95% significance level demonstrate the validity of the experiments.

4.2 University Email Hypernetwork Dataset

In what follows, we use a university email hypernetwork dataset [20,41]. The dataset contains email messages delivered to users via the university email system over six separate semesters. The email user population is a mix of students, faculty members, staff, and "affiliates" (a category including postdocs, visiting scholars, and alumni) in the university. Every email record is composed of date, time, sender, and list of recipients. Out of privacy and security concerns, the contents of email messages are discarded and the email addresses are encrypted. However, in this email system, we are allowed to access an email user table that describes the personal information of each user, namely occupation, birth, gender, home country, postal code, years at the university, academic department (for student and faculty), division (for student only), and dormitory building (for student only). Email messages from each of the six semesters are treated as a separate dataset. Each person is treated as a vertex in the hypernetwork, and all the persons that appear in one email are regarded as a set of vertices in a hyperlink. The personal information from every email user is regarded as observed features for each vertex. The average number of nodes for each dataset is 67,736, and the average number of hyperlinks is 253,469. We obtain positive data and negative data using similar scheme as last subsection. Detailed information of the six datasets is listed in Table 3.

We execute HPLSF under each of the six datasets respectively. Ob-Model, Ex-Model-MDS, and Ex-Model-LFRM are used for comparison. In Ob-Model, we use all the accessible user information as observed features. In Ex-Model-LFRM,

Table 3. The overview of the university email hypernetwork datasets

Time	#Vertices	#Hyperlinks
Semester1	57,328	340,717
Semester2	61,451	248,009
Semester3	65,946	131,448
Semester4	73,040	458,273
Semester5	77,256	163,930
Semester6	71,396	178,435

Table 4. The hyperlink prediction accuracy (%) using HPLSF, Ob-Model, Ex-Model-MDS, and Ex-Model-LFRM under the university email hypernetwork datasets

Time	HPLSF	Ob-Model	Ex-Model-MDS	Ex-Model-LFRM
Semester1	$\mathbf{88.7 \pm 0.9}$	82.8 ± 1.1	79.2 ± 1.8	72.5 ± 1.3
Semester2	$\mathbf{89.4 \pm 0.6}$	81.2 ± 2.0	78.7 ± 1.0	73.2 ± 3.0
Semester3	$\mathbf{92.4 \pm 1.0}$	83.2 ± 0.9	83.6 ± 1.4	77.6 ± 3.3
Semester4	$\mathbf{89.7 \pm 1.8}$	80.4 ± 1.7	74.3 ± 2.2	67.7 ± 1.8
Semester5	$\mathbf{90.1 \pm 1.2}$	81.3 ± 1.3	82.3 ± 1.8	79.6 ± 1.1
Semester6	$\mathbf{87.1 \pm 1.2}$	79.4 ± 2.2	83.0 ± 1.5	79.6 ± 1.3

variational inference is used to learn the parameter in the latent feature relational model as described in [26]. The latent feature dimension p is determined using the same scheme as last subsection. SVM^{struct} [14] is applied for training and prediction. In all the methods, we employ the 10-fold cross validation scheme to achieve the average prediction accuracy and list them in Table 4. Table 4 indicates that the prediction accuracy of HPLSF outperforms all the baselines under all the datasets. HPLSF achieves significantly higher accuracy than Ob-Model, which demonstrates that exploring "latent" social features are helpful and necessary for hyperlink predictions because social features are not always observable. Meanwhile, the accuracy of Ex-Model-MDS is much higher than Ex-Model-LFRM under almost all datasets, which implies that HPLSF designs a better way to explore latent features in hypernetworks. Pairwise t-tests at 95% significance level demonstrate the validity of the experiments.

4.3 IT Company Email Hypernetwork Dataset

In what follows, an email hypernetwork dataset collected from a large information technology and electronics company [18,17] is employed to evaluate the proposed HPLSF framework. The dataset contains the complete record, as drawn from the company's servers, of email communications among 30,328 employees from 2006 to 2008.The employees in the company are located in 289 different offices around 50 states in United States and collectively comprise about one quarter of the company's employee population. Each email record comprises the timestamp, sender, lists of receipients, and the size of the message. Privacy laws and corresponding company policies preclude the collection of the content of messages. However, some personal information for each employee is accessible from the HR department of the company, namely work years in the company, employee's job function, office location code, the state of the office, and employee's group ID. Email messages from each of the three years are treated as a separate dataset. Each person is regarded as a vertex in the hypernetwork while all people appearing in one email message are regarded as vertices associated with the hyperlink. The personal information for each employee obtained from HR department is treated as observed features. Detailed information of the three datasets is shown in Table 5.

Table 5. The overview of the IT company email hypernetwork datasets

Year	#Vertices	#Hyperlinks
2006	30,328	992,382
2007	27,134	1,074,507
2008	27,134	473,756

Table 6. The hyperlink prediction accuracy (%) using HPLSF, Ob-Model, Ex-Model-MDS, and Ex-Model-LFRM under the IT company email hypernetwork datasets

Year	HPLSF	Ob-Model	Ex-Model-MDS	Ex-Model-LFRM
2006	$\mathbf{87.4 \pm 0.8}$	74.8 ± 0.2	76.5 ± 2.2	73.7 ± 0.9
2007	$\mathbf{85.1 \pm 0.6}$	81.3 ± 1.3	78.4 ± 2.4	75.3 ± 2.0
2008	$\mathbf{86.7 \pm 1.1}$	77.5 ± 1.3	81.4 ± 1.7	76.8 ± 1.3

We run HPLSF on each of the three datasets sequentially and obtain the hyperlink prediction results. To compare with the proposed algorithm, the three baselines are used as in last section. In Ob-Model, all available personal information from the HR department is used as each employee's observed features. In Ex-Model-LFRM, we still use variational inference to learn the parameter for the LFRM. For all the methods, 10-fold cross validation scheme is employed to calculate the average prediction accuracy.

The results listed in Table 6 show that the proposed HPLSF achieves higher accuracy than any other baseline methods. HPLSF performs much better than Ob-Model, establishing that "latent" social features are helpful for hyperlink predictions. Meanwhile, Ex-Model-MDS still performs better than Ex-Model-LFRM on all the three datasets, which demonstrates again that the latent feature learning scheme designed in HPLSF is better than LFRM under the hypernetwork scenario. Pairwise t-tests at 95% significance level demonstrate the validity of the experiments.

5 Conclusion

In this paper, we propose a link prediction framework for hypernetworks, which is, to the best of our knowledge, the first hyperlink prediction work. The framework, named HPLSF, aims to predict whether a hyperlink exists among a set of vertices in a hypernetwork by leveraging not only observed features but also latent features. By designing a feature-embedding technique, we address the artbitrary-sized hyperlink cardinality challenge in hypernetwork setting. Because observed features and latent features are not necessarily independent of each other, we generalize a structural SVM rather than simply combining the results obtained from classifiers using each of the two types of features. The experimental results under three large email hypernetworks from diverse sources demonstrates the effectiveness of HPLSF.

References

1. Backstrom, L., Leskovec, J.: Supervised random walks: predicting and recommending links in social networks. In: WSDM 2011 (2011)
2. Barros, R.C., Basgalupp, M.P., de Carvalho, A., Freitas, A.A.: A survey of evolutionary algorithms for decision-tree induction. IEEE Trans. SMC 42(3), 291–312 (2012)
3. Bautu, E., Kim, S., Bautu, A., Luchian, H., Zhang, B.-T.: Evolving hypernetwork models of binary time series for forecasting price movements on stock markets. In: IEEE Evolutionary Computation 2009 (2009)
4. Bichot, C.-E., Siarry, P.: Graph Partitioning: Optimisation and Applications. Wiley (2011)
5. Bonachich, P., Holdren, A., Johnston, M.: Hyper-edges and multidimensional centrality. Social Networks 26(3), 189–203 (2004)
6. Cox, T.F., Cox, M.A.A.: Multidimensional Scaling. Chapman and Hall (2001)
7. Gloor, P.A., et al.: Towards growing a coin in a medical research community. Procedia Social and Behavioral Sciences (2010)
8. Gao, S., Denoyer, L., Gallinari, P.: Temporal link prediction by integrating content and structure information. In: CIKM 2011 (2011)
9. Grangier, D., Melvin, I.: Feature set embedding for incomplete data. In: NIPS 2010 (2010)
10. Ha, J.-W., Eom, J.-H., Kim, S.-C., Zhang, B.-T.: Evolutionary hypernetwork models for aptamer-based cardiovascular disease diagnosis. In: GECCO 2007 (2007)
11. Hoff, P.D.: Modeling homophily and stochastic equivalence in relational data. In: NIPS 2007 (2007)
12. Hoff, P.D., Raftery, A.E., Handcock, M.S.: Latent space approaches to social network analysis. J. American Statistical Association 97, 1090–1098 (2001)
13. Hopcroft, J., Khan, O., Kulis, B., Selman, B.: Natural communities in large linked networks. In: SIGKDD 2003 (2003)
14. Joachims, T., Finley, T., Yu, C.J.: Cutting-plane training of structural svm. Machine Learning 77(1), 27–59 (2009)
15. Kang, F., Jin, R., Sukthankar, R.: Correlated label propagation with application to multi-label learning. In: CVPR 2006 (2006)
16. Kim, S., Kim, S.-J., Zhang, B.-T.: Evolving hypernetwork classifiers for microrna expression profile analysis. In: IEEE Evolutionary Computation 2007 (2007)
17. Kleinbaum, A.M.: Organizational misfits and the origins of brokerage in intra-firm networks. Administrative Science Quarterly 57, 407–452 (2012)
18. Kleinbaum, A.M., Stuart, T.E.: Inside the black box of the corporate staff: Social networks and the implementation of corporate strategy. Strategic Management Journal (2013)

19. Kondor, R., Jebara, T.: A kernel between set of vectors. In: ICML 2003 (2003)
20. Kossinets, G., Watts, D.J.: Empirical analysis of an evolving social network. Science (2006)
21. Liben-Nowell, D., Kleinberg, J.: The link prediction problem for social networks. In: CIKM 2003 (2003)
22. Lichtenwalter, R.N., Lussier, J.T., Chawla, N.V.: New perspectives and methods in link prediction. In: SIGKDD 2010 (2010)
23. Macskassy, S.A., Provost, F.: Classification in networked data: A toolkit and a univariate case study. JMLR 8, 935–983 (2007)
24. McFee, B., Lanckriet, G.: Metric learning to rank. In: ICML 2010 (2010)
25. McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: Homophily in social networks. Annual Review of Sociology 27(1), 415–444 (2001)
26. Miller, K.T.: Bayesian nonparametric latent feature models. Ph.D. Thesis, University of California, Berkeley (2011)
27. Miller, K.T., Griffiths, T.L., Jordan, M.I.: Nonparametric latent feature models for link prediction. In: NIPS 2009 (2009)
28. Neville, J., Jensen, D.: Leveraging relational autocorrelation with latent group models. In: SIGKDD Workshop 2005 (2005)
29. Newman, M.: The structure and function of complex networks. SIAM Review 45(1), 167–256 (2003)
30. Newman, M.E.J.: Modularity and community structure in networks. Proceedings of the National Academy of Sciences 103(23), 8577–8582 (2006)
31. Palla, G., Derenyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature 435(7043), 814–818 (2005)
32. Pothen, A.: Graph partitioning algorithms with applications to scientific computing. Technical Report, Norfolk, VA (1997)
33. Shi, J., Malik, J.: Normalized cuts and image segmentation. TPAMI 22(8), 888–905 (2000)
34. Sun, L., Ji, S., Ye, J.: Hypergraph spectral learning for multi-label classification. In: SIGKDD 2008 (2008)
35. Tang, L., Liu, H.: Relational learning via latent social dimensions. In: SIGKDD 2009 (2009)
36. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector learning for interdependent and structured output spaces. In: ICML 2004 (2004)
37. Wang, C., Satuluri, V., Parthasarathy, S.: Local probabilistic models for link prediction. In: IEEE ICDM 2007 (2007)
38. Xie, L., Gu, N., Li, D., Cao, Z., Tan, M., Nahavandi, S.: Concurrent control chart patterns recognition with singular spectrum analysis and support vector machine. Computers and Industry Engineering 64(1), 280–289 (2013)
39. Xie, L., Li, D., Simske, S.J.: Feature dimensionality reduction for example-based image super-resolution. Journal of Pattern Recognition Research 2, 130–139 (2011)
40. Xu, Y., Ping, W., Campbell, A.: Multi-instance metric learning. In: ICDM 2011 (2011)
41. Xu, Y., Rockmore, D.: Feature selection for link prediction. In: PIKM 2012 (2012)
42. Zhou, D., Huang, J., Schölkopf, B.: Learning with hypergraphs: Clustering, classification, and embedding. In: NIPS 2006 (2006)
43. Zhu, J.: Max-margin nonparametric latent feature models for link prediction. In: ICML 2012 (2012)

Extracting Opinionated (Sub)Features from a Stream of Product Reviews

Max Zimmermann[1], Eirini Ntoutsi[2], and Myra Spiliopoulou[1]

[1] Otto-von-Guericke University of Magdeburg, Magdeburg 39106, Germany
{max.zimmermann,myra}@iti.cs.uni-magdeburg.de
[2] Ludwig-Maximilians-University of Munich, Germany
ntoutsi@dbs.ifi.lmu.de

Abstract. We propose a stream mining method that learns opinionated product features from a stream of reviews. Monitoring the attitude of customers towards products is a field of much interest, but the products themselves may come in and out of the market. We rather investigate which (implicit) features of the products are important for the customers, and monitor how customer attitude towards such features evolves. To this purpose, we use a two-level stream clustering algorithm that extracts features and subfeatures from an opinionated stream, and couple it with dedicated feature-specific classifiers that assess the polarity of each extracted (sub)feature. We evaluate our method on a stream of reviews and we elaborate on how changes in the arrival rate of features (drift) affects algorithm performance.

Keywords: stream mining, product feature extraction, opinion mining.

1 Introduction

We investigate the problem of monitoring sentiment in product reviews. Opinion mining on products is widespread. We rather focus on identifying and monitoring the product *features*, which are frequently mentioned by the consumers. We propose a stream mining method that learns the features and assesses the polarity mostly associated with each feature. As the stream progresses, our method adjusts the features, forgetting unpopular ones and recognizing emerging ones, and monitors the change of their polarity over time - and thus the attitude of the consumers towards product features.

Opinion monitoring over streams of reviews has gained momentum in the last years. Stream learners have been proposed to extract opinions from streams, detecting drifts and bursts [3,4,15]. Opinion monitoring responds to the fact that the attitude of people may change over time. Hence, learning from a stream of opinionated reviews contributes to better decision making for the customers and to better estimation of product popularity for the product owner.

One thread of opinion mining concentrates on identifying product features and assessing the sentiment associated to them [9,16]. A *feature* is an implicit property, as e.g. in “This camera lens is cheap” (lens as property of the camera).

Monitoring of product features is a natural extension of static learning; it is useful for two reasons. First, products enter and exit the market, but the popularity of some features remains, e.g. the lens of *any* camera, the battery lifetime for *any* laptop. Second, features that suddenly become popular call for the producers' attention: if many reviews on the "charge device" of different cameras emerge, this indicates that customers have become interested in that feature.

In this work, we study opinion monitoring for features of products [1]. Our stream mining approach encompasses (a) product feature extraction and adaption to new features, (b) feature polarity learning and (c) filtering of unimportant reviews. For product feature extraction (item a) we define a hierarchy of polarized (sub)features, so that different levels of granularity on the product properties are found. We adapt this feature hierarchy over time, incorporating new features and eliminating unpopular ones, so that changes in the emphasis paid to features by customers is captured in the model. For feature polarity learning (item b), we train one sentiment classifier per feature, so that effects of polysemous words can be minimized ("heavy" is negative for a laptop but may be positive for a lens). Finally, (item c) we introduce the notion of *important review*, as review that is similar to many others and can serve as representative. We consider only important reviews, so that we capture general trends and omit outliers, while we gain also in efficiency. For feature extraction from important reviews we extend our earlier method TStream [18], which monitors topics on streams and detects novel topics.

The rest of the paper is organized as follows: In Section 2, we discuss related work. In Section 3, we describe the basic concepts, and then we introduce our approach. Section 4 contains our experiments on a real dataset, where we simulate drifts and bursts. The last Section concludes our study.

2 Related Work

Relevant to our work are studies on sentiment analysis over streams, on feature extraction from a stream of opinionated documents and on stream clustering.

Sentiment Analysis over Streams. One of the first approaches on sentiment analysis over a stream was proposed by Silva et al. [15]: stream learning starts with a small seed of labeled documents, upon which a classification rules learner is trained. The seed is gradually expanded with new relevant documents. We also use a small seed for learning, but we train a classifier for each feature.

Bifet and Frank [3] investigate sentiment classification on a stream of tweets; they consider unbalanced classes with drifts and shifts in the class distribution, under the requirement of quick response under memory constraints. Closest to our approach is their follow-up framework [4] that consists of (i) a twitter filter to convert tweets into TF-IDF vectors, (ii) an adaptive frequent itemset miner that stores the frequency of the most frequent terms and (iii) a change detector

[1] We use the terms 'feature' and 'property' as synonyms, to denote an implicit property that must be extracted from the reviews with text (stream) mining methods.

that explores changes in the frequency distribution of the items. The framework monitors changes in the frequency of words. We also propose a framework for stream learning over opinionated documents, but our objective is to first identify the features and subfeatures of the products we study, and then to assess the polarity of the features (using a classifier for each (sub)feature), thereby taking feature specific words into account and also consider that features may emerge and disappear as the stream progresses.

Feature Extraction from Reviews. Feature extraction and monitoring from a stream is a new subject. For feature extraction on a static set of reviews, Liu identifies four research subtopics [10], of which the identification of frequent nouns and of noun phrases are closest to our research.

Long et al. [11] extract core words for an aspect, compute their frequencies, estimate their distance to other words and use it to acquire further words related to the aspect. Zhu at al. [17] consider the frequency of terms that contain other terms. Mukherjee et al. [13] extract features and relationships among them: for feature extraction, they consider all nouns. In contrast, we suppress very frequent nouns with the help of TF-IDF weighting. Moghaddam and Ester [12] want to find multi-part noun phrases like "LCD display"; they use TF-IDF weighting of nouns with non-stopword stems at document- and paragraph level and they apply Apriori to find frequent noun combinations. We also aim to find multi-word terms, but use two-level clustering instead; this allows us to identify also refinements of features. All above methods are static; our approach also captures emerging features and gradually forgets features that are no longer important.

Stream Clustering. We can distinguish two types of stream clustering methods. Methods of the first type summarize the stream and maintain summaries online; clustering is an offline step. An early approach of this type is Clustream [1], more recent ones include DenStream [5]. We adhere to the second type of stream clustering, where the clusters are updated as new data instances arrive. An early approach of this type appeared in [7], the text stream clustering algorithm in [2] adheres to this type. For text stream clustering in the current work, we build upon our earlier method TStream [18], which is specialized in detecting new topics from bursts of news and in accumulating them to a fixed set of clusters.

3 Extracting and Maintaining Polarized Features

We monitor a stream of product reviews, from which we extract a two-level hierarchy of product features, assess the polarity assigned to the features by the people who write the reviews, and identify changes in feature polarity over time.

Our approach is designed for streams of product reviews, where each review refers to a single feature of the product. The stream itself, though, covers a variety of features of the different products. The requirement of one feature per review may look a bit restrictive at first. However, we are mainly interested in the few dominant products features that customers focus on, especially when they decide to write only *brief* reviews. Long appraisals of content (e.g. for books) are beyond our scope. Long reviews that address many features of the same product can be split into short sentences. Our framework would currently consider these sentences as independent; exploiting their correlations is issue for future work.

Briefly, our framework works as follows. We process the stream in batches of fixed size at timepoints $t_0, t_1, \cdots, t_i, \cdots$. Since the batch size is fixed (to a constant we denote as *streamSpeed*), the timepoints are not equidistant. On this stream, we perform text stream clustering, by building upon our algorithm TStreams that derives topics and subtopics from a stream of news [18]. TStreams partitions the first batch of reviews into K_g clusters at the first hierarchy level – from these clusters we extract the *product features*. It then partitions each global cluster into K_l local clusters –from these we extract the *product subfeatures*. As new batches arrive, TStreams pushes reviews down the hierarchy, while keeping reviews that do not fit any cluster into containers. When containers are filled, the hierarchy is rebuilt. We extend TStreams to detect and process only "important" reviews, which are, informally, similar to many other reviews and can thus serve as representatives. For each global and local cluster, we learn a polarity classifier. All classifiers are initialized on a first batch of labeled reviews and then extended through label propagation. When a cluster is rebuilt, its dedicated classifier is also re-learned. The framework is depicted in Algorithm 1 and described in detail in subsections 3.2 and 3.3, after introducing definitions and notation.

3.1 Definitions and Notation

The objective of our framework is to learn features and their polarity. To do so, we first extract from each batch of the reviews' stream the "important" reviews.

Definition 1 (Review Importance). *Let r be a review and R a dataset containing it. We define the "importance of r with respect to R" as the number of reviews in R that have r among their k nearest neighbors, whereby the reviews are weighted on their "age" (cf. Def. 2 below).*

$$importance(r, R) = \sum_{r_i \in R} age(r_i) \cdot isRevNeighbour(r, r_i, R)$$

where:
$$isRevNeighbour(r, r_i, R) = \begin{cases} 1, r \in NN(k, r_i, R) \\ 0, \text{otherwise} \end{cases}$$

and $NN(k, r_i, R)$ is the set of k-nearest neighbors of r_i in R; we use cosine similarity as similarity function.

Hence, a review is important with respect to some dataset R. This dataset is a cluster of the two-level hierarchy. Within R, r is imporant if it appears among the k nearest neighbors of *many recent* reviews and can thus serve as their representative. Recency is regulated by the concept of *age*:

Definition 2 (Review Age). *The age of a review r is the average age of all words w_i contained in r: $age(r) = \frac{1}{|r|} \sum_{w_i \in r} exp\left(-\lambda \cdot (t - t_{w_i})\right)$
where t is the current timepoint, t_{w_i} is the time of the most recent review that contains w_i and $\lambda \in \Re$ is a decay factor.*

On the basis of Defs. 1 & 2, we rank reviews on importance and apply a review importance threshold β to select the most important ones. These constitute a dataset R, from which we derive a feature space of nouns F_R. We use the feature space to vectorize the reviews (with TF-IDF) and then perform clustering on R. Then, extending the definition of “topic” in [18], we define a “polarized feature” as a cluster centroid with an associated polarity:

Definition 3 (Polarized Feature). *Let R be a dataset of reviews labeled on polarity, and let F_R be the vector space learned upon R (through TF-IDF). Let $c \subset R$ be a cluster. The “polarized feature” represented by c consists of:*

- *the centroid $\hat{c} = \prec w_1, w_2, \ldots, w_{|F_R|} \succ$, where w_i is the average TF-IDF weight of keyword noun $k_i \in F_R, i = 1 \ldots |F_R|$.*
- *the polarity label $c^{polarity}$, defined as the majority class label within c*

TStreams builds a two-level hierarchy [18]. We extend it by learning the clusters of the 1st level from the important reviews only. The same is done at the 2nd hierarchy level: within each “global cluster”, the unimportant reviews are removed, the local feature space is computed and the cluster is partitioned into subclusters (“local clusters”). The centroid of a local cluster, associated with the majority class label in it is then a polarized sub-feature (by Def. 3).

Not all arriving reviews can fit into the existing hierarchy. We inherit from [18] the notion of *novelty* for a document with respect to the existing clusters:

Definition 4 (Review Novelty). *Let r be a new review. Let θ be a set of clusters extracted from a dataset R. Let F_R be the vector space derived from R (through TF-IDF). Given a similarity threshold $\delta \in [0,1]$, r is novel with respect to θ if its cosine similarity to the closest cluster centroid is less than δ, where the cosine similarity depends on the feature space ($cosine_{F_R}$).*

Novel reviews are maintained separately in containers. As in [18], we associate the 1st hierarchy level with a *global container*, which accommodates reviews that are too far from all centroids of all global clusters. Each such cluster is further associated with a *local container*, which accommodates reviews that are close to its centroid but far from all centroids of its subclusters (local clusters). To decide when to re-cluster the contents of one global cluster only or the whole set of global clusters, we monitor the *novelty degree of the stream*, which we implement on the basis of the size of the containers (Def. 5 comes from [18]):

Definition 5 (Stream Novelty). *Let θ be a set of clusters. and let $\mathcal{Z}$ be the container associated with θ; it contains all those reviews that are novel with respect to θ, according to Def. 4. Given a size threshold parameter σ, $\mathcal{Z}$ exhibits novelty towards θ if: $|\mathcal{Z}| \geq \sigma$.*

When enough novel reviews have arrived, the model is *updated* through reclustering at the first or second level. In-between the updates, the model is *adapted* by incorporating the non-novel reviews. For the adaption, we take the *importance* of the reviews into account, as defined in Def. 1, subject to the review importance threshold β. For our experiments, we have set $\beta = 0.6$.

3.2 Extracting an Initial Hierarchy of Polarized (Sub)Features

To extract the hierarchy of (sub)features from an initial set of opinionated reviews R (line 1) we build upon TStreams [18]. Global clusters (features) are extracted by applying clustering over the entire initial set of reviews; a total of K_g global clusters is extracted. To derive the local clusters (subfeatures), clustering is applied again over the sets of reviews corresponding to each of the global clusters. This way, a unique TF-IDF feature space is built for each global cluster and the corresponding local clusters are extracted from this feature space. A total of K_l local clusters are extracted for each global cluster.

To learn the polarity of the derived (sub)features, as expected for Def. 3, we train a Multinomial Naive Bayes (MNB) classifier Δ for each global and local cluster of the hierarchy (line 1), based on the initial reviews that are in these clusters and thus support the corresponding (sub)features (polarized centroids, cf. Def. 3). The choice for MNB is motivated by the good performance reported in [14]. However, rather than training one global classifier (as in [14]) on reviews that may be heterogeneous in content, we train local classifiers on the homogeneous reviews inside each cluster. Note that the hierarchy of (sub)features evolves over time based on the new coming reviews and the ageing of the old ones, the maintenance of the hierarchy is discussed in Section 3.3.

To vectorize the reviews, we use the Bag-of-Words model, but we consider only adjectives and adverbs, because, according to [10], these parts of speech express best the subjective opinions of the authors.

3.3 Adapting the Evolving Feature Hierarchy and the Feature Polarities

New reviews might cause smooth or drastic changes at both the hierarchy (sub)features and their associated classifiers. Smooth changes call for adaptation whereas drastic changes require re-building of (part of) the hierarchy. The decision depends upon the novelty of the incoming reviews.

More specifically, upon the arrival of a new review r (line 6-20) from the stream, our method works as follows:

(a) Review novelty check and novelty accumulation. We first check whether the new review r is novel (line 9) w.r.t. the global clusters (features) of the hierarchy (cf. Def. 4). If so, r is propagated to the global container $\mathcal{Z}$ (line 19) where novel reviews are stored. Otherwise, r fits to an existing global cluster c_g, i.e. it supports the cluster's polarized feature. Then, either r fits to some local cluster under c_g (line 12-14) or it is assigned to the local container $\mathcal{Z}_{c_g}$ (line 16) [18].

Algorithm 1: Feature-Sentiment Extraction

```
   Input : initial seed $R$, stream $S$, set of parameters $\mathcal{L}$
 1 t ← 0; $\Theta \leftarrow extractPolarizedHierarchyAndClassifiers(R, \mathcal{L})$
 2 $importanceBookKeepingOfReviews(\Theta, \mathsf{t}, \lambda, k)$
 3 batch ← first $streamSpeed$ reviews from $S$
 4 while batch do
 5     t ← t +1
 6     for $i$=1 to |batch| do
 7         $currentReview$ ← $i^{th}$ position in batch
 8         $Cg \leftarrow findMostProximalGlobalCluster(currentReview, \delta_g, \Theta)$
 9         if $Cg$ is not null then
10             $updateCentroid(currentReview, Cg)$
11             $Cl \leftarrow findMostProximalLocalCluster(currentReview, \delta_l, Cg)$
12             if $Cl$ is not null then
13                 $updateCentroid(currentReview, Cl)$
14                 $assignLabel(currentReview, \Delta_{Cl})$
15             else
16                 $assignToContainer(\mathcal{Z}_{Cg}, currentReview)$
17                 $assignLabel(currentReview, \Delta_{Cg})$
18         else
19             $assignToContainer(\mathcal{Z}^{\Theta}, currentReview)$
20             $assignLabel(currentReview, \Delta^{\Theta}_{default})$
21     if $|\mathcal{Z}^{\Theta}| > \sigma_g$ then
22         $\mathcal{Z}^{\Theta}$ ← add $n$ latest important reviews
23         $\Theta \leftarrow extractPolarizedHierarchyAndClassifiers(\mathcal{Z}^{\Theta}, \mathcal{L})$
24     else
25         for $i$=1 to $K_g$ do
26             if $|\mathcal{Z}^{\Theta}_{Cg_i}| > \sigma_l$ then
27                 $\mathcal{Z}^{\Theta}_{Cg_i}$ ← add $n$ latest important reviews of $Cg^{\Theta}_i$
28                 $relearnClusters(Cg^{\Theta}_i, \mathcal{Z}^{\Theta}_{Cg_i}, \mathcal{L})$
29                 $relearnClassifiers(\Delta^{\Theta}_{Cg_i}, \mathcal{L})$
30     $importanceBookKeepingOfReviews(\Theta, \mathsf{t}, \lambda, k)$
31     $removeUnimportantReviews(\Theta, \beta)$
32     $updateClusterCentroids(\Theta)$
33     $storePolarizedHierarchy(\Theta, \mathsf{t})$
34     batch ← replace content of batch by the next $streamSpeed$ reviews from S
```

Table 1. Set of Parameters $\mathcal{L}$

Parameter	Definition
K_g, K_l	number of global, respectively local clusters
σ_g, σ_l	global, resp. local novelty threshold
δ_g, δ_l	global, resp. local similiarity threshold

Parameter	Definition
λ	decay constant
β	importance threshold
n	# important reviews to relearn
streamSpeed	# reviews per batch
k	# nearest neighbors

(b) Review assignment. A new review r that is not novel is assigned to its most proximal global and then local cluster (line 8 & 11). This means that r is associated with the feature and subfeature described by these clusters. The centroids of the associated clusters are updated by the content of r (line 10 & 13). We assess the polarity of r by invoking the classifier for the local cluster, to which r is assigned (line 14).

If r is assigned to a global cluster but not to a local one, the classifier of the global cluster is invoked (line 17). In case r is novel and it does not fit to the existing hierarchy, the default classifier is applied which is learned upon the whole dataset (line 20). The default classifier is the most generic one, whereas as we traverse the hierarchy the classifiers become more specific and thus, intuitively, better capture the sentiment of the associated reviews.

(c) Importance book-keeping. The importance score of each review (cf. Def. 1) is updated (line 30), since the score is affected by ageing of the words and by changes in the neighborhoods (due to the arrival and ageing of other reviews). Reviews that are not (no more) important are removed (line 31). Moreover, the updating of the importance of reviews implies updates in the centroids of the (sub)features (line 32) (cf. Def. 3), since old important reviews might be removed whereas new reviews might now be considered important.

Note that there is no need to update the importance of all the reviews in the hierarchy after the arrival of a new review. We do need to update the importance of the reviews for the cluster where this review has been assigned to since the inverse kNNs might change due to the addition of the new review. For the rest of the reviews though, change in the importance can be triggered only due to the natural ageing of the keywords and we need to update them once per timepoint. Recall that more than one review might arrive per timepoint, described by the *streamSpeed* parameter.

To facilitate the ageing computations in the importance formula (cf. Def. 1), we maintain for each cluster in the hierarchy a hashmap containing the words that appear in the cluster reviews, their frequency in the cluster and the last timestamp where each word has been observed in the cluster. This information is adequate for computing the ageing of each keyword in the cluster, while the hashmap entries are easily maintained as new reviews are assigned to the cluster and old anymore non-important reviews are removed as outdated. The inverse kNN queries are also not a bottleneck since they are restricted within each cluster and moreover, only the important reviews within a cluster contribute to their computation. As already mentioned, non-important reviews are removed from the cluster. In the experiments, we show that the consideration of only important reviews has a big effect on the runtime of our method (cf. Figure 6).

(d) Stream novelty check and model updating. When enough novel reviews have been accumulated in the containers, the hierarchy is rebuilt totally or partially (line 23 & 28) so as to discover new (sub)features and forget outdated ones. In particular, we rebuild the complete hierarchy if the size of the global container $\mathcal{Z}$ exceeds the novelty threshold σ (cf. Def. 5) (line 21). If only a local container is filled, only the corresponding global cluster is re-partitioned (line 26-28). For reclustering, we use the novel reviews and the n most recent of the old important reviews (line 22 & 27). Thus, we ensure that both new words and still popular old words are incorporated to the updated (sub)features. This step builds upon model updating in TStreams [18], extending it with the maintenance of the most important reviews.

(e) Updating the classifiers. When a set of reviews is re-clustered, a new classifier must be trained for each new cluster. We have the option of using only the initial seed for training, and the option of considering all reviews in the clusters but with the derived polarity labels. In our experiments, we use the latter option.

When the whole hierarchy is re-built (line 23), except for the new classifiers for the global and local clusters, the default classifier must also be trained. To this end, all the reviews in the hierarchy are considered.

4 Experiments

To evaluate our feature extraction method we use a real dataset of product reviews [8], from which we generate through instance permutation three streams with different properties. For polarity learning, we compare our one-classifier-per-cluster method to a static classifier learned once on an initial part of the stream and to an adaptive classifier that updates the model after each batch.

We selected all parameters experimentally. The similarity thresholds used for review importance are set to $\delta_g = 0.6$, $\delta_l = 0.8$ (similarity at 2nd level is more restrictive), while $\beta = 0.6$. The novelty thresholds for the containers are $\sigma_g = 100$, $\sigma_l = 15$, enforcing reclusterings inside a global cluster instead of rebuilding the whole hierarchy. The ageing factor λ is set to 0.5, the number of nearest neighbors $k = 4$. Batch size (*streamSpeed*) is equal to 50 reviews; the initialization batch contains 100 reviews. The number of reviews to relearn is $n = 2 \times streamSpeed$. All results are the average of 10 runs of our algorithm.

4.1 Datasets

The product reviews dataset of [8] contains 540 reviews on 9 products, where each review refers to one (implicit) product feature, from a total of 38 features.

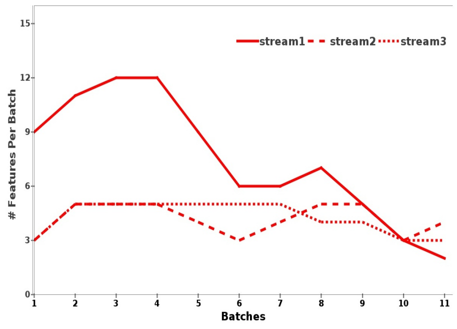


Fig. 1. # observed features per batch

Setting	stream1			stream2			stream3		
$K_g\_K_l$	a	b	c	a	b	c	a	b	c
2_2	3	0	250	3	0	250	3	0	250
4_6	2	0	350	2	1	350	2	1	350
4_9	2	0	350	2	1	350	2	1	350
6_4	2	1	400	2	1	450	1	1	400
6_6	2	1	450	1	1	450	1	1	400
7_2	1	1	400	1	1	450	1	1	500
8_8	1	0	500	1	1	500	1	0	500

Fig. 2. a: # full reclusterings, b: # partial reclusterings, c: # seen reviews when first full reclustering was invoked

Most features are mentioned in 9 to 30 reviews. From this dataset we derived three streams (cf. Figure 1) by sorting the reviews regarding their related features as described bellow; also we filter reviews which are associated to features that occur less than 9 times accross the dataset. Stream 1 (solid line) delivers all 38 features to the learner within the first 220 reviews. Stream 2 (dashed line) delivers only 6 features in the first 100 reviews, 8 further features are in the reviews 101 - 200 and so on; the last feature is seen around review 520. Stream 3 goes like Stream 2 but delivers a different selection of features per batch. The exact number of observed features per batch is depicted in Figure 1. Stream 1 covers a larger number of features per batch compared to streams 2 & 3. Since a feature corresponds to a (sub)cluster, we expect that the feature extraction method will achieve better results on stream 1 if the number of (sub)clusters is high. For streams 2 & 3, we expect a better performance since the number of features per batch is almost constant over time. Although the dataset is small, it allows us to experiment with different forms of evolution.

Figure 2 shows how the arrival of reviews triggers reclusterings. The first column contains the number of global clusters K_g and of subclusters below each global one, K_l . For each stream, column a depicts the number of reclusterings at the 1^{st}-level of the hierarchy (full reclusterings: they affect all clusters); column b counts the reclusterings at the 2^{nd}-level (partial reclusterings: they affect one global cluster each time). Column c depicts the number of reviews seen before the first full reclustering. A late first full reclustering and a low number of full reclusterings are indicators of good adaption to change and also of a good performance of our algorithm. similarity thresholds δ_g, δ_l remain constant. As we can see, our algorithm performs better for settings with more clusters in total (6_4 vs 2_2) or with many global clusters, even if the number of local clusters is low (7_2 vs 4_6).

4.2 Evaluating the Feature Extraction Method

We evaluate the feature extraction method on the "purity" of the (sub)features it extracts: purity is high if the number of features covered by each cluster is low. We first define the purity of a local cluster c_l inside a global cluster c_g as the percentage of the reviews supporting the most frequent feature in c_l w.r.t. all reviews in the cluster, weighting the reviews on importance (cf. Def. 1):

$$localPurity(c_l) = \frac{\sum_{r \in df_{c_l}} importance(r, c_g)}{\sum_{r \in c_l} importance(r, c_g)} \tag{1}$$

where where df_{c_l} is the set of reviews in c_l that are related to the majority feature w.r.t. all reviews in c_l.

Next, we aggregate the *localPurity*() values of all subclusters of global cluster c_g into *globalPurity*() – a normalized sum, weighted by the number of features covered by each subcluster c_l and by the number of reviews contained in c_l:

$$globalPurity(c_g) = \frac{\sum_{c_l} localPurity(c_l) \cdot coveredFeatures(c_l) \cdot |c_l|}{coveredFeatures(c_g) \cdot |c_g|} \tag{2}$$

Finally, we aggregate into the *average weighted purity* of a clustering Θ:

$$avgWPurity(\Theta) = \frac{1}{|\Theta|} \sum_{c \in \Theta} globalPurity(c) \tag{3}$$

In Figure 3, we study how the average weighted purity changes with the arrival of reviews for streams 1 (left) & 3 (right) under different number of global and local clusters; we skip stream 2, because all values are very similar to stream 3. Note that the arrival of reviews affects the number of features known and remembered at each timepoint (cf. Figure 1). Therefore, it affects the number of features described by each subcluster and consequently the cluster purity. Each curve corresponds to a different number of global clusters (first number: 2, 4, 6, 7, 8) and local clusters (second number: 2, 4, 6, 8); for example, the solid line with circles at the junctions is labeled 7_2 and corresponds to 7 global clusters, each one containing 2 subclusters. The points at each curve do not correspond to exact *avgWPurity* values but to averages over the batch size; for example, the first point corresponds to the average of *avgWPurity* over the first 100 reviews.

Figure 3 shows that purity increases with the number of global clusters. For all streams, the purity decreases after clustering the initial set of 100 reviews, but then stabilizes and slightly increases again. We also see that the ratio of global-to-local clusters has different effects for the different streams. For example, the purity of the setting 2_2 is better for stream 2 & 3 than for stream 1. An explanation is that streams 2 & 3 see less features at the arriving batches in comparison to stream 1 (cf. Figure 1). So, some of the features seen at the beginning of streams 2 & 3 are later forgotten, hence the clusters accommodating them are used to describe new, emerging features. Thus, under streams 2 & 3, the full set of features can be described with less clusters since at each timepoint not all features are present.

We juxtapose the peaks of the curves (between 300 and 400 reviews for streams 2 & 3, between 200 and 300 for stream 1) in Figure 3 with the number of reclusterings shown in Figure 2. We see that the Stream 1 curves with early peaks

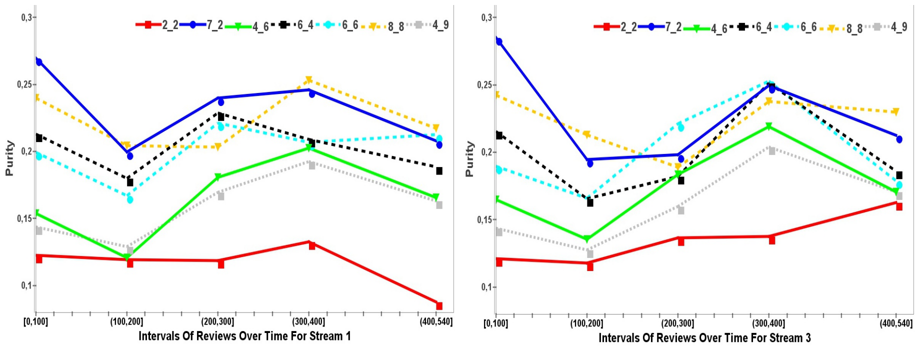


Fig. 3. *avgWPurity* for streams 1 and 3: higher values are better. Each curve $c_g c_l$ corresponds to a configuration of c_g global and c_l local clusters. Each point in the curve corresponds to the average over the *avgWP*() values for the reviews seen within the interval depicted in the horizontal axis.

(6_4,6_6 and 7_2) correspond to partial reclusterings. So, partial reclusterings lead to peaks on purity and thus improve model quality.

For the setting 6_4 on stream 3, we show in Table 2 the labels of the subclusters and how they change over time. The first column is the identifier of the global cluster, the rest of the columns correspond to batches. We use the notation c (C) to denote a local cluster c occurring within a global cluster C. The upper part of the table contains the first seven batches, whereas the remaining three batches are depicted in the lower part of the table. As the stream progresses, there are subclusters with empty labels (e.g. , c:1 (C:0) at batch 200). These correspond to obsolete features, i.e. features that have not received any new reviews from the stream in the recent past; the old reviews supporting them aged and vanished. We see that some clusters, as c:4 (C:0), are very stable, covering the same feature (for c:4 (C:0) it is "support") throughout. According to [8], this feature is mentioned only for three products, hence c:4 (C:0) nicely extracts the feature independently of the products. A different example is the feature "odor" that is supported by reviews from only one product, which is very different from the others. This feature is not a global one; it is accommodated in local cluster c:21 (C:20). Thus, the two-level hierarchy captures the semantics of the features in the reviews, whereby it is better to allow for many clusters at the 1st level; this comes at no cost, since clusters remain empty if there are no features to be covered.

4.3 Evaluation of the Polarity Learning Method

For the evaluation of our one-classifier-per-cluster method we use accuracy, taking as ground truth the actual scores given to each review by the users. There are six scores: -3 (most negative polarity),- 2, -1, 1, 2, 3 (most positive polarity). We compare our method to two baselines, the *StaticBaseline* and the *DynamicBaseline*. The *StaticBaseline* is a single global classifier trained over the first

Table 2. Stream 3, setting 6_4: labels of the local clusters $(c : x)$ within global clusters $(C : y)$ for each batch

Name	100	150	200	250	300	350	400
C:0	c:1/size; c:2/install; c:3/price; c:4/support;	c:1/size; c:2/install; c:3/price; c:4/look;	c:1/; c:2/; c:3/; c:4/support;	c:1/; c:2/; c:3/; c:4/support;	c:1/; c:2/; c:3/; c:4/support;	c:1/; c:2/; c:3/; c:4/support;	c:1/; c:2/; c:3/; c:4/support;
C:5	c:6/power; c:7/power; c:8/battery; c:9/working;	c:6/power; c:7/power; c: 8/sound; c:9/ working;	c:6/power; c:7/power; c:8/sound; c:9/working;	c:6/software; c:7/power; c:8/sound; c:9/working;	c:6/software; c:7/; c:8/sound; c:9/;	c:6/software; c:7/; c:8/control; c:9/;	c:6/software; c:7/; c:8/control; c:9/;
C:10	c:11/use; c:12/sound; c:13/ headphones; c:14/ headphones;	c:11/screen; c:12/sound; c:13/quality; c:14/ headphones;	c:11/use; c:12/sound; c:13/ headphones; c:14/ headphones;	c:11/use; c:12/sound; c:13/ headphones; c:14/ headphones;	c:11/use; c:12/; c:13/look; c:14/ headphones;	c:11/use; c:12/; c:13/look; c:14/ installation;	c:30/install; c:31/use; c:32/setup; c:33/software;
C:15	c:16/screen; c:17/sound quality; c:18/battery; c:19/quality;	c:16/screen; c:17/sound quality; c:18/battery; c:19/quality;	c:16/screen; c:17/sound quality; c:18/battery; c:19/;	c:16/screen; c:17/diaper pail; c:18/ installation; c:19/;	c:16/screen; c:17/diaper pail; c:18/ installation; c:19/;		
C:20	c:21/odor; c:22/screen; c:23/diaper pail; c:24/control;	c:21/odor; c:22/screen; c:23/sound; c:24/control;	c:21/odor; c:22/LCD; c:23/sound; c:24/control;	c:21/odor; c:22/ installation; c:23/sound; c:24/control;	c:21/works; c:22/ installation; c:23/sound; c:24/control;	c:21/size; c:22/ installation; c:23/sound; c:24/control;	c:21/size; c:22/ installation; c:23/size; c:24/control;
C:25	c:26/works; c:27/ adjustment; c:28/works; c:29/ adjustment;	c:26/ battery life; c:27/ adjustment; c:28/works; c:29/ adjustment;	c:26/ battery life; c:27/ adjustment; c:28/works; c:29/ adjustment;				

Name	450	500	540
C:39	c:40/interface; c:41/; c:42/control; c:43/;		
C:44	c:45/battery; c:46/iTunes; c:47/battery; c:48/;	c:45/install; c:46/software; c:47/setup; c:48/;	c:64/install; c:65/bluetooth; c:66/control; c:67/screen;
C:54	c:55/; c:56/storage; c:57/price; c:58/;	c:55/; c:56/sound; c:57/; c:58/;	c:55/; c:56/sound; c:57/; c:58/;
C:59	c:60/; c:61/battery; c:62/; c:63/;		

100 reviews from the stream. The *DynamicBaseline* trains an initial classifier over the first 100 reviews. Then, as in prequential evaluation [6], the next batch of reviews is used first to evaluate the classifier and then for learning based on the true review labels. The accuracy values are shown in Figure 4 for stream 3.

In Figure 4, we see that in the early phases of the stream, our approach outperforms the baselines for most of the settings. This indicates that the use of dedicated feature-specific classifiers is a good alternative to a global, generic classifier *as long as* the features are well-separated.Towards the end of the stream, the accuracy of our approach deteriorates, although it never drops much below the baselines. An explanation is that the last batches of stream 3 contain many more features than can be accommodated in the clusters, thus affecting also the performance of the dedicated classifiers.

What polarity values are depicted for the extracted features? In Figure 5, we show the feature names and polarities (setting 4_4) under stream 1. The polarity is captured as a shade of gray (dark stands for positive). We see the evolution of the features across the vertical axis (time) and from left to right, since larger cluster identifiers correspond to later clusters. For example, observe cluster C:35, which first describes the feature "interface" (a positively perceived product property). Its polarity changes at the next timepoint, indicating that this

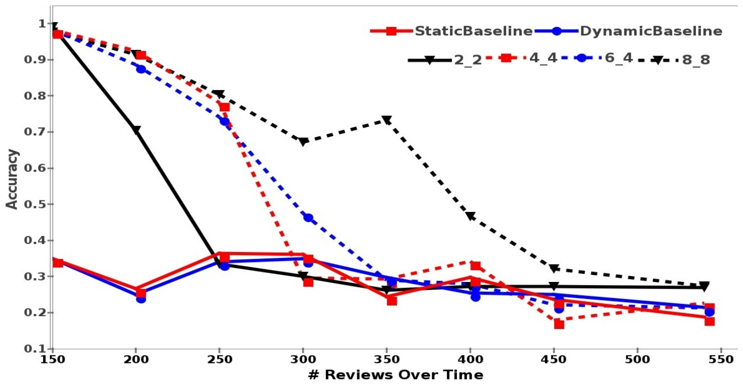


Fig. 4. Accuracy over time for stream 3 settings of our method

feature is not perceived positively anymore. Then, the feature becomes obsolete and is replaced by another one, "price", which is also perceived rather negatively in this dataset. The cluster C:35 dies out at timepoint 7.

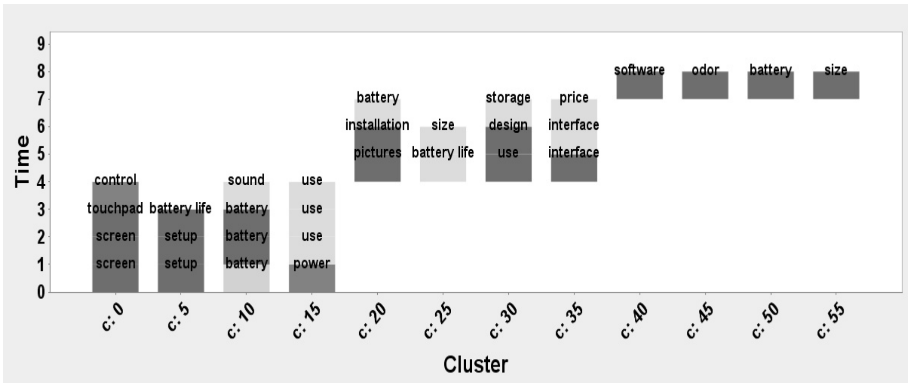


Fig. 5. Label & Polarity evolution per cluster on Stream 1 for 4 global clusters and 4 local ones: The polarity is captured as a shade of gray, where dark stands for positive

Finally, we studied how the filtering of unimportant reviews affects the runtime of our approach. In Figure 6, we vary the threshold β that determines how many reviews will be considered as important. As expected, the more selective the importance filter is (higher β values), the lower the execution time. Thus, the runtime of our algorithm is regulated by the value of β rather than the size of the stream.

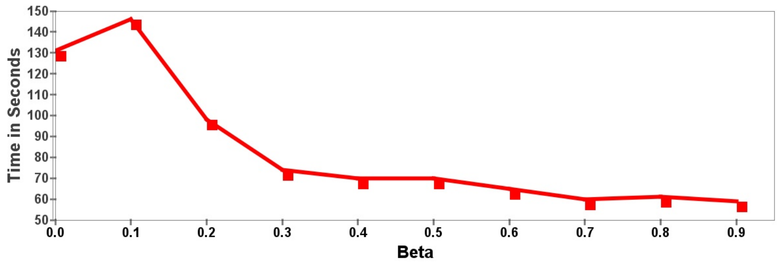


Fig. 6. Execution time of our method for different β values

5 Conclusion

We presented a method for the discovery of opinionated product features in a stream of product reviews. We use a stream clustering algorithm to extract and maintain a two-level hierarchy of product features. For each feature, we learn a dedicated classifier that predicts and monitors feature polarity over time. Our method can capture new features in the data and forget obsolete ones, whereupon it rebuilds the model in a parsimonious way, re-learning only locally (within some clusters) whenever possible. To avoid noise and decrease execution time, we concentrate only on "important" reviews for learning, i.e. we sample the arriving reviews, selecting those that are similar to many others and can thus serve as representatives for them.

We evaluated our approach on a stream of product reviews with respect to the quality of the extracted features and to the accuracy of the dedicated classifiers. Our experiments show that our method performs well, especially when the number of clusters in the two-level hierarchy is large enough to accommodate all features. Clusters that accommodate no features remain empty, hence specifying a large number of clusters does not incur substantial overhead. Further, the execution time of our method is governed by the number of *important* reviews it considers and not by the size of the whole stream.

As a next step, we want to investigate the interplay between cluster purity (how many features are covered by a cluster) and classifier performance, so that classifiers for features with very few and possibly heterogeneous reviews are avoided. Moreover, we want to evaluate our method on reviews that cover more than one feature, where there is one feature per sentence. Also we want to study heuristics to adapt the number of global/local clusters dynamically based on the heterogeneity of the reviews which are detected as novel. Further, we want to investigate ways of making the classifiers adaptive, instead of re-learning them from scratch whenever some global cluster is re-built.

Acknowledgments. Work of Max Zimmermann was funded by the German Research Foundation project SP 572/11-1 "IMPRINT: Incremental Mining for Perennial Objects".

References

1. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: VLDB (2003)
2. Aggarwal, C.C., Yu, P.S.: A framework for clustering massive text and categorical data streams. In: SDM (2006)
3. Bifet, A., Frank, E.: Sentiment knowledge discovery in twitter streaming data. In: Pfahringer, B., Holmes, G., Hoffmann, A. (eds.) DS 2010. LNCS, vol. 6332, pp. 1–15. Springer, Heidelberg (2010)
4. Bifet, A., Holmes, G., Pfahringer, B.: MOA-TweetReader: Real-time analysis in twitter streaming data. In: Elomaa, T., Hollmén, J., Mannila, H. (eds.) DS 2011. LNCS, vol. 6926, pp. 46–60. Springer, Heidelberg (2011)
5. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: SDM (2006)
6. Gama, J.A., Sebastião, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, pp. 329–338. ACM, New York (2009)
7. Guha, S., Meyerson, A., Mishra, N., Motwani, R., O'Callaghan, L.: Clustering data streams: Theory and practice. IEEE TKDE 15(3), 515–528 (2003)
8. Hu, M., Liu, B.: Mining and summarizing customer reviews. In: Proc. of the Tenth ACM SIGKDD Int'l. Conference on Knowledge Discovery and Data Mining, KDD 2004, pp. 168–177. ACM, New York (2004)
9. Liu, B.: Opinion mining and summarization. Tutorial at the World Wide Web Conference (WWW), Beijing, China (2008)
10. Liu, B.: Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers (2012)
11. Long, C., Zhang, J., Zhut, X.: A review selection approach for accurate feature rating estimation. In: Proc. of the 23rd Int'l. Conference on Computational Linguistics, COLING 2010. Association for Computational Linguistics (2010)
12. Moghaddam, S., Ester, M.: Opinion digger: an unsupervised opinion miner from unstructured product reviews. In: Proc. of the 19th ACM Int'l. Conference on Information and Knowledge Management, CIKM 2010. ACM (2010)
13. Mukherjee, S., Bhattacharyya, P.: Feature specific sentiment analysis for product reviews. In: Gelbukh, A. (ed.) CICLing 2012, Part I. LNCS, vol. 7181, pp. 475–487. Springer, Heidelberg (2012)
14. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up?: sentiment classification using machine learning techniques. In: Proc. of the ACL 2002 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, pp. 79–86. Association for Computational Linguistics, Stroudsburg (2002)
15. Silva, I.S., Gomide, J., Veloso, A., Meira Jr., W., Ferreira, R.: Effective sentiment stream analysis with self-augmenting training and demand-driven projection. In: Proc. of the 34th Int'l. ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 475–484. ACM, New York (2011)
16. Zhang, Y., Shen, D., Baudin, C.: Sentiment analysis in practice. Tutorial at the IEEE International Conference on Data Mining, Vancouver, Canada (2011)
17. Zhu, J., Wang, H., Tsou, B.K., Zhu, M.: Multi-aspect opinion polling from textual reviews. In: Proc. of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, pp. 1799–1802. ACM (2009)
18. Zimmermann, M., Ntoutsi, E., Siddiqui, Z.F., Spiliopoulou, M., Kriegel, H.-P.: Discovering global and local bursts in a stream of news. In: Proc. of the 27th Annual ACM Symposium on Applied Computing, SAC 2012. ACM (2012)

Author Index