ECML PKDD SEP 07 11 TEM BER 2015 PORTO, PORTUGAL

Proceedings of the 4th Workshop on

New Frontiers in Mining Complex Patterns (NFMCP 2015)

Editors

Michelangelo Ceci

Corrado Loglisci

Giuseppe Manco

Elio Masciari

Zbigniew W. Ras

New Frontiers in Mining Complex Patterns (NFMCP 2015)

The analysis of complex data represents the new frontier in data mining and knowledge discovery. There are several emerging technologies and applications where complex patterns can be extracted: examples are blogs, event or log data, medical data, spatio-temporal data, social networks, mobility data, sensor data and streams. The abundance, variety and velocity of data poses new challenges which can be hardly coped with traditional data mining techniques. This asks for new contributions which allow for efficiently identifying patterns and enable effective decision making.

The Fourth International Workshop on New Frontiers in Mining Complex Patterns (NFMCP 2015) was held in Porto in conjunction with the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2015) on September 7, 2015. It was aimed at bringing together researchers and practitioners of data mining and knowledge discovery, interested in the advances and latest developments in mining complex data. The workshop is establishing a premiere event with this goal.

This book features a collection of revised and significantly extended versions of papers accepted for presentation at the workshop. These papers went through a rigorous review process and each paper has bee reviewed by at least three reviewers. The individual contributions of this book illustrate advanced data mining techniques which preserve the informative richness of complex data and allow for efficient and effective identification of complex information units present in such data.

The book is composed of four parts and a total of 16 chapters.

Part I focuses on **Mining Data Stream Streams** and it consists of 5 chapters. Chapter 1 explores how different local and global tree-based approaches for multi-target regression compare in the streaming setting. Chapter 2 proposes a hardware-based parallel algorithm for frequent itemset mining on data streams which exploits the notions of systolic trees and Landmark window. Chapter 3 focuses on the analysis of event logs, in order to discover (overlapping) communities of resources and track the evolution of these communities over consecutive time windows. Chapter 4 proposes a technique for adapting the generation of a bagging classifier on a data stream to memory requirement. Finally, Chapter 5 investigates the task of determining changes which are regularly repeated over time.

Part II analyses issues posed by **Classification** in presence of complex data. It consists of three chapters. Chapter 6 is an experimental study of the properties that influence the performance Roughly Balanced Bagging on complex data. In Chapter 7, the authors address the problem of classifying log traces describing business process executions, with regards to potential security breaches. Chapter 8 presents a novel algorithm for mining redescriptions (i.e., different descriptions

of subsets of elements in the data) based on multi-label Predictive Clustering Trees.

Part III presents algorithms and applications where complex patterns are discovered from **heterogeneous data**. It contains four chapters. Chapter 9 focuses on inductive reasoning and studies the problem of detecting analogical patterns (i.e., the matching characteristics of two different items) and to enhance them through generalization techniques. Chapter 10 addresses the problem of automatic vehicle identification based on audio information. Chapter 11 focuses on graph mining and proposes a new technique for defining a graph kernel which embeds the underlying graphs by a random sample of their spanning trees. Chapter 12 presents an approach to mining heterogeneous information networks applied to a task of categorizing customers linked in a heterogeneous network of products, categories and customers.

Finally, Part IV focuses on **Time Series and Sequences**. Again, it contains four chapters. Chapter 13 proposes an application of sequential pattern mining to predict future occurrences of some pre-specified queries, thus enabling the maintenance of their indices on-demand. Chapter 14 presents a transductive learning framework for classifying multivariate sequences. The framework exploits PCA to build a graph of the correlations between multivariate sequences, and exploits both labeled and unlabeled sequences. Chapter 15 presents a string representation for hourly foreign exchange data and evaluates the performance of a trading strategy derived from it. Finally, chapter 16 focuses on recurrence plots, and studies the effect of a video compression algorithm on the classification performance of recurrence plots and the related time series.

We would like to thank all the authors who submitted papers for publication in this book and all the workshop participants and speakers. We are also grateful to the members of the program committee and external referees for their excellent work in reviewing submitted and revised contributions with expertise and patience. We would like to thank Jerzy Stefanowski for his invited talk on "Adaptive ensembles for evolving data streams - combining block-based and on-line solutions". A special thank is due to both the ECML PKDD Workshop Chairs and to the ECML PKDD organizers who made the event possible. We would like to acknowledge the support of the European Commission through the project MAESTRA - Learning from Massive, Incompletely annotated, and Structured Data (Grant number ICT-2013-612944) and of the Italian ministry of the education and research through the project Cybersecurity (Grants PON03PE_00032_1, PON03PE_00032_2 and PON03PE_00032_3).

July 30, 2015

Michelangelo Ceci,
Corrado Loglisci,
Giuseppe Manco,
Elio Masciari,
Zbigniew Ras.

Organization

Program Chairs

Michelangelo Ceci	University of Bari "Aldo Moro", Bari, Italy
Corrado Loglisci	University of Bari "Aldo Moro", Bari, Italy
Giuseppe Manco	ICAR-CNR, Rende, Italy
Elio Masciari	ICAR-CNR, Rende, Italy
Zbigniew Ras	University of North Carolina, Charlotte, USA & Warsaw University of Technology, Poland

Program Committee

Nicola Barbieri	Yahoo Research Barcelona
Elena Bellodi	ENDIF-University of Ferrara
Petr Berka	University of Economics, Prague
Michelangelo Ceci	Università di Bari, Italy
Ivica Dimitrovski	Faculty of Computer Science and Engineering
Bettina Fazzinga	CNR-ICAR Italy
Stefano Ferilli	Universita' di Bari
Joao Gama	University Porto
Thomas Gärtner	Fraunhofer Institute IAIS
Dino Ienco	IRSTEA
Dragi Kocev	Jozef Stefan Institute
Corrado Loglisci	University of Bari
Giuseppe Manco	CNR-ICAR, Italy
Elio Masciari	CNR- ICAR, Italy
Mirco Nanni	KDD-Lab ISTI-CNR Pisa
Ruggero G. Pensa	University of Torino, Italy
Gianvito Pio	University of Bari "Aldo Moro"
Chiara Pulice	UNICAL, Italy
Zbigniew Ras	University of North Carolina, Charlotte
Domenico Redavid	Artificial Brain
Jerzy Stefanowski	Poznań Univeristy of Technology, Poland
Herna Viktor	University of Ottawa
Alicja Wieczorkowska	Polish-Japanese Institute of Information Technology
Wlodek Zadrozny	UNCC

Additional Reviewers

Fassetti, Fabio
Flesca, Sergio
Furfaro, Filippo
Guarascio, Massimo
Ritacco, Ettore
Viktor, Herna
Ziembinski, Radoslaw

Table of Contents

INVITED TALK

Adaptive ensembles for evolving data streams - combining block-based and on-line solutions 1
Jerzy Stefanowski

Data Stream Mining I

Tree-based Approaches for Multi-target Regression on Data Streams 2
Aljaž Osojnik, Pance Panov and Saso Dzeroski

A Hardware-Based Approach for Frequent Itemset Mining in Data Streams 14
Lazaro Bustio, Raudel Hernandez Leon, René Cumplido, Claudia Feregrino and José Manuel Bande Serrano

Data Stream Mining II

Discovering and Tracking Organizational Structures in Event Logs 26
Annalisa Appice, Marco Di Pietro, Claudio Greco and Donato Malerba

Intelligent Adaptation of Ensemble Size in Data Streams Using Online Learning 38
M. Kehinde Olorunnimbe, Herna Viktor and Eric Paquet

Mining Periodic Changes in Complex Dynamic Data through Relational Pattern Discovery 50
Corrado Loglisci and Donato Malerba

Classification

Applicability of Roughly Balanced Bagging for Complex Imbalanced Data 62
Mateusz Lango and Jerzy Stefanowski

Classifying traces of event logs on the basis of security risks 74
Bettina Fazzinga, Sergio Flesca, Filippo Furfaro and Luigi Pontieri

Redescription mining with multi-label Predictive Clustering Trees 86
Matej Mihelčić, Sašo Džeroski, Nada Lavrač and Tomislav Smuc

Mining complex data

Cross-domain Generalization by Analogy 98
Fabio Leuzzi and Stefano Ferilli

Spectral Features for Audio Based Vehicle Identification 110
Alicja Wieczorkowska, Elzbieta Kubera, Tomasz Slowik and Krzysztof Skrzypiec

Probabilistic Frequent Subtree Kernels 122
Pascal Welke, Tamas Horvath and Stefan Wrobel

Heterogeneous networks decomposition andweighting with text mining heuristics 134
Jan Kralj, Marko Robnik-Šikonja and Nada Lavrač

Sequences

AIDA: an application of sequential pattern mining 146
Mirjana Mazuran, Matteo Simoni and Letizia Tanca

Multivariate Sequence Analysis with Transductive Classification 156
Zhao Xu and Koichi Funaya

Evaluating a simple string representation for intra-day foreign exchange prediction 166
Simon Cousins and Blaz Zlicar

Time Series Classification using Compressed Recurrence Plots 178
Thilo Michael, Stephan Spiegel and Sahin Albayrak

Adaptive ensembles for evolving data streams - combining block-based and on-line solutions

Jerzy Stefanowski

Institute of Computing Science, Poznań University of Technology,
60-965 Poznań, Poland

Abstract. The rapid development of the information technology facilities collecting big data sets which cause challenges for their storage and processing. In particular, one faces the difficulties with massive volumes of data in the form of data streams. Compared to the static environments mining data streams implies new requirements for algorithms, such as constraints on memory usage, restricted processing time, and one scan of incoming examples. The other critical issue is reacting to concept drifts. These requirements make mining evolving data streams a complex problem which asks for new dedicated solutions. As standard approaches are not appropriate for learning classifiers from changing data streams, several new algorithms have been introduced in the last years. Ensembles are among the most often studied classifiers. Due to their modularity, they provide a natural way of adapting to changes by modifying their structure, either by retraining ensemble members, replacing old component classifiers with new ones, or updating rules for aggregating component predictions. Most of current ensembles do not contain a drift detector and react to changes in the adaptive way. They can be further divided into block-based and online approaches.

This talk starts with a general overview of the current data stream ensembles. Then, we study differences between block-based and on-line ensembles with respect to: different reaction to various types of drifts, time and memory requirements and strategies to learn component classifiers. We hypothesize that it is still possible to develop new types of hybrid ensembles that combine the most beneficial properties of these both types of approaches. In the next part of this talk we present experiences from using such two algorithms, recently developed in our group.

The first algorithm, called Accuracy Updated Ensemble (AUE), is a more block-based oriented proposal. It includes elements of incremental updating of component ensembles and a new aggregation rule. Its experimental evaluation shows that it provides the better classification accuracy than other state-of-the-art algorithms, also with acceptable time and memory usage.

The AUE ensemble is, then, generalized into its completely incremental version, called the On-line Accuracy Updated Ensemble. Its experimental evaluation shows advantages with respect to the faster reaction to several types of drifts. Finally, we discuss open research directions for constructing ensembles from complex data streams.

Tree-based Approaches for Multi-target Regression on Data Streams

Aljaž Osojnik[1,2], Panče Panov[1], and Sašo Džeroski[1,2,3]

[1] Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia
[2] Jožef Stefan IPS, Jamova cesta 39, Ljubljana, Slovenia
[3] CIPKeBiP, Jamova cesta 39, Ljubljana, Slovenia
e-mail: {aljaz.osojnik, pance.panov, saso.dzeroski}@ijs.si

Abstract. Single-target regression is a classical data mining task that is popular both in the batch and in the streaming setting. Multi-target regression is an extension of the single-target regression task, in which multiple continuous targets have to be predicted together. Recent studies in the batch setting have shown that, global approaches that predict all of the targets at once tend to outperform local approaches that predict each target separately. In this paper, we explore how different local and global tree-based approaches for multi-target regression compare in the streaming setting. Specifically, we apply a local method based on the FIMT-DD algorithm and propose a novel global method, named iSOUP-Tree-MTR. Furthermore, we present an experimental evaluation that is mainly oriented towards exploring the differences between the local and global approaches. Finally, the results confirm the findings from the batch setting that the global approaches generally outperform the local ones.

1 Introduction

A common approach to complex data mining tasks is to transform them into simpler tasks, which have known solutions. This problem transformation approach has been used to address predictive tasks, such as the multi-label classification and multi-target regression tasks. A multi-label classification task can thus be transformed into a collection of binary classification tasks, while a multi-target regression task can be decomposed into several single-target regression problems.

There are, however, methods that forego the reduction to simpler tasks and tackle the complexity head-on. Specifically, in the case of multi-target regression, methods that consider and predict all of the continuous targets at once have received considerable coverage in the literature [17,12]. Almost exclusively, though, these methods have been introduced in the batch setting.

Recently, however, the streaming setting is becoming more and more prominent, in large part due to the ever increasing presence of Big Data problems and applications. The constraints of the streaming setting lend themselves conveniently towards addressing several of the characteristics of Big Data, i.e., the "V"s of Big Data. Specifically, streaming methods are generally exposed to Velocity – the high speed of arrival of data, Volume – potentially unbounded number of data instances and Variability – potential changes in the data itself.

Methods that address multi-target regression in the streaming setting are few and far between, especially those that predict all of the targets at once. In this paper, we present a new tree-based approach, named iSOUP-Tree-MTR, capable of addressing multi-target regression in this manner. We compare it to the simple problem transformation approach of using streaming single-target tree-based methods and show that the iSOUP-Tree.MTR method has superior performance. Additionally, we explore the performance of ensembles, e.g., online bagging [14], when using the iSOUP-Tree-MTR method as a base learner.

The structure of the paper is as follows. First, we present the background and related work (Sec. 2). Next, we present several tree-based approaches for multi-target regression on data streams (Sec. 3). Furthermore, we present the research questions and the experimental design (Sec. 4). Finally, we conclude with a discussion of the results (Sec. 5), conclusions, and further work (Sec. 6).

2 Background and Related Work

In this section, we define the multi-target regression task and present the constraints of the streaming context. Additionally, we briefly review the state-of-the art in multi-target regression, both in the batch and in the streaming setting.

Multi-target Regression. In essence, we can look at the multi-target regression task as an extension of the single-target regression task. In the later, only one continuous variable needs to be predicted. The multi-target regression (MTR) task deals with predicting multiple numeric variables simultaneously, or, formally, with making a prediction $\hat{y}$ from $\mathbb{R}^n$, where n is the number of targets for a given instance $\boldsymbol{x}$ from an input space X. To categorize the different approaches to MTR we use the nomenclature introduced by Silla and Freitas [11] for the task of hierarchical multi-label classification. The task of simultaneous prediction of all targets at the same time (the *global* approach) has been considered in the batch setting by Struyf and Džeroski [17]. In addition, Appice and Džeroski [1] proposed a method for stepwise induction of multi-target model trees.

Data Streams. Unlike the batch context, where a fixed and complete dataset is given as an input to a learning method, the streaming context presents several constraints that a stream learning method must consider. The most relevant are [2]: (1) the examples arrive sequentially; (2) there can potentially be arbitrarily many examples; (3) the distribution of examples need not be stationary; and (4) after an example is processed it is discarded or archived. The fact that the distribution of examples is not presumed to be stationary means that methods should be able to detect and adapt to changes in the distribution (*concept drift*).

Multi-target Regression on Data Streams. In the streaming context, some recent work has already addressed the single- and multi-target regression task. Ikonomovska et al. [10] introduced an instance-incremental streaming tree-based single-target regressor (FIMT-DD), which utilizes the Hoeffding bound. This

work was later extended to for multi-target regression (FIMT-MT) [9]. However, both of these methods had the drawback of ignoring nominal input attributes. There has been some theoretical debate whether the use of the Hoeffding bound is appropriate [15], however, a recent study by Ikonomovska et al. [8] has shown that in practice the use of the Hoeffding bound produces good results. Additionally, Shaker et al. [16] introduced an instance-based system for classification and regression (IBLStreams), which can be, in principle, used for multi-target regression.

3 Tree-based Methods for Multi-target Regression on Data Streams

Generally, the quickest way of solving a complex task, such as multi-target regression, is to transform it into simpler tasks that have known solutions. In the case of multi-target regression, specifically, this is achieved by training a regressor for each of the targets separately, essentially resulting in a collection/ensemble of regressors. The other option for addressing the multi-target regression task is to produce a regressor which gives predictions for all of the targets at once.

To differentiate these approaches we refer to them as local and global, respectively [11]. Specifically, a method that uses one regressor per target is using the *local approach*, while a method that uses one regressor to predict all of the targets at once is using the *global approach*. Recent studies show, that in the batch case, the global approaches outperform the local ones [12]. Global methods tend to (implicitly) exploit dependencies between the targets.

In this section, we present several tree-based methods for multi-target regression, which utilize the local approach, as well as the global approach. Tree-based methods are often used, as they generally provide good results in terms of predictive performance, while also yielding interpretable models. Finally, we present a baseline method that can be viewed as both local and global and is highly relevant to the methods introduced below.

3.1 A Local Approach to MTR

One of the best known single-target tree-based regressors in the stream setting is the FIMT-DD method [10]. It is based on the Hoeffding bound, which allows for the generalization of observations from small samples to the underlying distribution. Similarly to Hoeffding trees used for classification [4], FIMT-DD uses the Hoeffding bound to determine the best splits of the resulting decision tree.

We have re-implemented the FIMT-DD method in the Java-based MOA stream-mining framework [3] and extended it to use adaptive models in the leaves, similarly to Duarte et. al [5]. Specifically, each leaf of the tree contains a perceptron. The perceptron is a linear function of the values of the input attributes $\boldsymbol{x}$ that produces the prediction, i.e., $\hat{y} = \boldsymbol{w} \cdot \boldsymbol{x} + b$, where $\boldsymbol{w}$ and b are a learned weight vector and a constant, respectively.

In the original implementation, the perceptron was always used to make the prediction. However, the adaptive model records the errors of the perceptrons and compares them to the errors of the mean target predictors, which predict the value of the target by computing the average value of the target over the examples observed in a given leaf. In essence, each leaf has two learners, the perceptron and the target mean predictor. The prediction of the learner that (at any given point in time) has the lower error is then used as the final prediction.

To monitor the errors, we use the faded absolute error which is calculated as $fMAE_{learner}(m) = \frac{\sum_{j=1}^{m} 0.95^{m-j}|\hat{y}(j)-y(j)|}{\sum_{j=1}^{m} 0.95^{m-j}}$, where m is the number of observed examples in a leaf, $\hat{y}(j)$ and $y(j)$ are the predicted and real value for the j-th example, respectively, and $learner \in \{perceptron, targetMean\}$. In essence, the faded error is weighted towards more recent examples. Intuitively, the numerator of the fraction is the faded sum of absolute errors, while the denominator is the faded count of examples. For example, the most recent (m-th) example contributes with a weight of 1, the previous example with weight 0.95, and the first example with weight 0.95^{m-1}. This places an emphasis on the more recent examples and generally benefits the perceptron, as we expect its errors to decrease as it learns the weight vector.

We have implemented a meta-learning method in MOA that creates a homogeneous ensemble of single-target regressors and combines their single-target predictions into a multi-target prediction in real-time to facilitate the use of FIMT-DD as a multi-target regressor. This combination of methods is referred to as the **Local FIMT-DD** method.

3.2 A Global Approach to MTR

As noted earlier, the global approach has been shown to yield better predictive performance in the case of tree-based methods in the batch setting. This has motivated the introduction of global tree-based methods for data streams, i.e., the FIMT-MT method introduced by Ikonomovska et al. [9]. FIMT-MT extends FIMT-DD by replacing the use of the variance reduction heuristic with the intracluster variance reduction heuristic, which captures some of the dependencies of the targets. One of the major downsides of the FIMT-MT method, however, is the fact that it completely ignores nominal input attributes.

We have extended FIMT-MT by adding the support for nominal input attributes. We gave also proposed the use of this extension to address other structured output prediction tasks, e.g., multi-label classification [13]. This new method is named **i**ncremental **S**tructured **Ou**tput **P**rediction **Tree** for MTR (**iSOUP-Tree-MTR**). As before, iSOUP-Tree-MTR is implemented in MOA.

In each leaf, the iSOUP-Tree method uses an adaptive multi-target model, which consists of a multi-target perceptron and a multi-target target mean predictor. Similarly to the single-target case, the multi-target perceptron produces the prediction vector as $\hat{\boldsymbol{y}} = W\boldsymbol{x} + \boldsymbol{b}$, where W is the weight matrix and $\boldsymbol{b}$ is the additive vector of constants. On the other hand, the multi-target target mean predictor computes the prediction as the mean value of each of the targets ob-

served at a given leaf. Individually, these learners can be seen as local, however, in conjunction with the tree building method, they constitute a global method.

For each target y_i the errors $fMAE^i_{perceptron}$ and $fMAE^i_{targetMean}$ are recorded and the decision which of the predictions to use is made for each variable separately. Formally, for each $i \in \{1, \ldots, n\}$ the prediction $\hat{y}^i_{perceptron}$ is used when $fMAE^i_{perceptron} < fMAE^i_{targetMean}$, otherwise we use $\hat{y}^i_{targetMean}$. This means that a final prediction $\hat{\boldsymbol{y}} = (\hat{y}^1, \ldots, \hat{y}^n)$ can be composed of some predictions made by the perceptron and some made by the target mean predictor.

Furthermore, we consider an ensemble of iSOUP-Tree base learners. Specifically, we use the bagging method for introducing diversity among the ensemble members. The bagging method for data streams was introduced by Oza et al. [14] and incorporates a probabilistic variation of how many times each given example is "seen" by each of the base learners. This combination of methods is referred in this paper as **iSOUP-Tree-MTR bagging**.

3.3 Baseline

An adaptive multi-target model is also used as a **baseline** regressor for the comparison of tree-based methods for multi-target regression on data streams, as it conveniently corresponds to both an ensemble of leaves using the local approach, as well as to a single leaf in the global iSOUP-Tree approach. In essence, the adaptive model corresponds to a tree-based model that is not allowed to grow, i.e., with leaves that are never split. The baseline is specifically implemented as a stripped down version of a single leaf node of an iSOUP-Tree-MTR.

4 Experimental Setup

In this section, we first present the experimental questions that we want to answer in this paper. Next, we discuss the evaluation measures used in the experiments and present the experimental methodology. Finally, we describe the datasets and conclude with the methods used in the experiments.

4.1 Experimental Questions

The first experimental question, that we wish to address in this paper, is the experimental comparison of local and global approaches. As the streaming context imposes several constraints on the learning process, it is not immediately clear whether the findings from the batch setting will be replicated in the streaming setting. While we are specifically using tree-based methods for multi-target regression on data streams, showing that the global approach increases predictive performance could also suggest that this may be generally true, i.e., applicable for other types of methods and structured outputs in the streaming setting.

When the number of targets is low, we expect the local approach to be competitive with the global approach in terms of time. However, when the number of

targets increases, we expect that the training of several distinct, even if simpler, models could take more time than the training of a single, more complex, model.

In a single-target study, Ikonomovska et al. [8] have shown no particular differences in the predictive performance of the basic method and the bagging method (therein referred to as FIMT-DD and OBag, respectively). In this work, we wish to investigate whether similar conclusions can be drawn in the multi-target case. To that end, we explore the differences in predictive performance between the iSOUP-Tree-MTR and iSOUP-Tree-MTR bagging methods. The time consumption of the bagging method will, naturally, be considerably higher, so any potential benefits must be evaluated through the lens of the additional use of resources.

4.2 Evaluation Measures and Experimental Methodology

From the experimental questions it is clear that we will need to observe several measures: two measures will be used to assess the predictive performance and another one to track the use of time. To assess the predictive performance, we will use the *mean absolute error* of a data sample D for each of the targets, $MAE^i = \frac{1}{|D|}\sum_{j=1}^{|D|} |\hat{y}^i(j) - y^i(j)|$, where $i \in \{1, \ldots, n\}$ indexes the targets. When applicable and for brevity we will, generally, consider the *average mean absolute error*, $\overline{MAE} = \frac{1}{n}\sum_{i=1}^{n} MAE^i$. It should be stressed that the average mean absolute error can only be considered when all of the targets are in the same approximate range, as targets with larger values would be over-represented in the average mean absolute error.

To evaluate the time consumption, we will consider the running time of the methods. Both the time consumption and the predictive performance measures are reported at intervals of 1000 examples. Specifically, we are using the *hold-out* approach to evaluating methods for mining data streams. This means that a *holdout set* (or *a window*) of fixed size is collected as enough examples accumulate, after which the predictions on the holdout set are used to calculate and report the evaluation measures. Following that, the model is updated with the collected examples and the process is repeated until all of the examples have been used. Each window, in essence, first takes the role of testing and then training set from the batch experimental setup, allowing for continuous monitoring of evaluation measures. In our case, the size of the window corresponds to the length of the measure reporting interval, i.e., 1000 examples.

4.3 Datasets

We have selected three datasets for our experiments, based on their size, increasing number of input and target attributes and the assumption of no concept drift, i.e., these datasets are generally considered as batch datasets.

The *Bicycles* dataset [6] is concerned with the prediction of demand for rental bicycles on an hour-by-hour basis. The 3 targets represent the predicted number of casual (non-registered) users, the predicted number of registered users and

Table 1: Datasets used in the experiments and their properties. N – number of instances, T – number of targets.

Dataset	Domain	N	Attributes	T
Bicycles [6]	service prediction	17379	12 numeric	3
EUNITE03	quality prediction	8064	29 numeric	5
SCM1d [18]	price prediction	9803	280 numeric	16

the total number of users for a given hour, respectively. In this case, the targets do not operate on the same range. Specifically, the third target is the sum of the previous two, so, naturally, it's range is different. For this dataset we report the mean absolute error for each target separately.

The *EUNITE03*[4] dataset was used for the competition at the 3rd European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems in 2003. The data describes a complex process of continuously manufactured glass products, i.e., the input attributes describe various influences which can or can not be changed by an operator, while the target attributes describe the glass quality. The targets are all reported on the same scale, so the use of the average mean absolute error is justified.

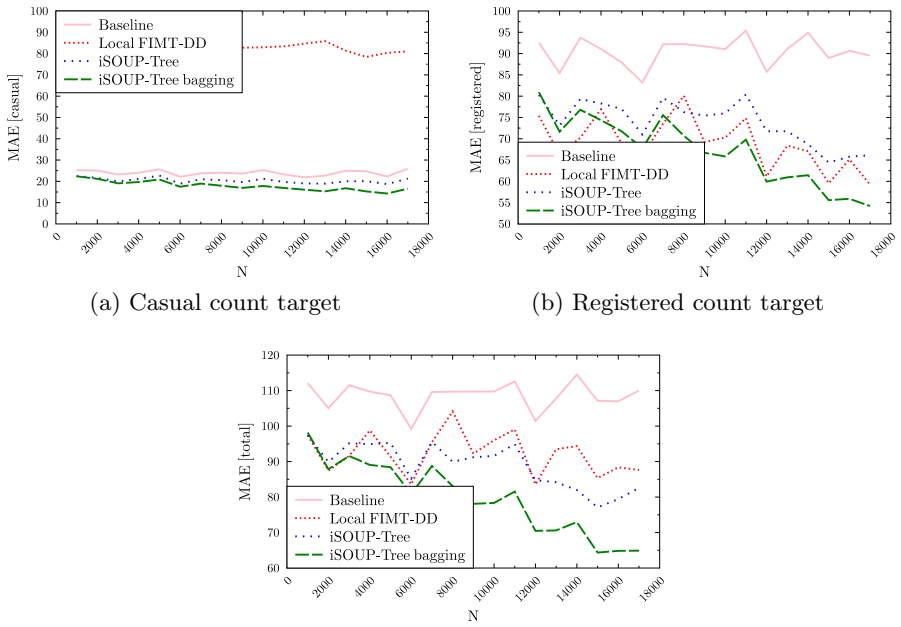
SCM1d is a dataset derived form the Trading Agent Competition in Supply Chain Management (TAC SCM) competition conducted in July 2010. The preparation (preprocessing) of the dataset is described by Xioufis et al. [18]. The data instances correspond to daily updates in a tournament – there are 220 days in each game and 18 games per tournament. The 16 targets are the predictions of the next day mean price for each of the 16 products in the simulation. The ranges of the targets are comparable among themselves, therefore, we will also use the average mean absolute errors on the SCM1d dataset.

The Bicycles dataset is available at the UCI Machine Learning Repository[5] and the SCM1d dataset is available at the Mulan multi-target regression dataset repository[6]. A summary of the datasets and their properties is shown in Tab. 1.

Notably, the Bicycle dataset exhibits strong seasonal effects, i.e., different parts of the concept are oversampled in different time periods. Since the datasets consist of measurements at different time points, the data instances are temporally ordered. This means that if we observe the data stream as is, the initial part of the dataset will contain examples which are severely oversampled from only one part of the input space. As data stream mining methods have no control over the arrival of data instances, the methods would, in the initial parts of the data stream, learn only part of the concept, i.e., the input attributes which exhibit seasonality would not be considered as relevant to the model building process, even though the seasonal concepts are different.

To avoid the overlearning of the potential seasonal concepts, we shuffled all of the datasets, i.e., we randomized the order of the data instances presented to

[4] `http://www.eunite.org/eunite/news/Summary%20Competition.pdf`
[5] `https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset`
[6] `http://mulan.sourceforge.net/datasets-mtr.html`



(a) Casual count target

(b) Registered count target

(c) Total count target

Fig. 1: The evolution of the mean absolute errors on the Bicycles dataset.

the learning method. If we used the datasets in their original representation, the results on the datasets with seasonality would show an increase in performance, while the learner is processing one of the seasons and then the performance would deteriorate, at the change of season.

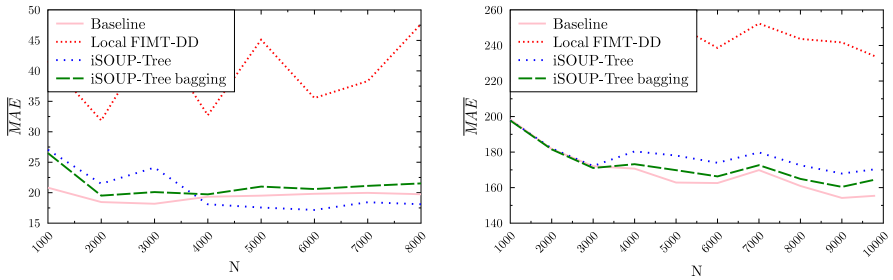
4.4 Compared Methods

For our experiments we consider the local and global tree-based methods described in Sec. 3. Specifically, we consider the multi-target perceptron as the baseline, the local FIMT-DD-based approach to MTR, and the global iSOUP-Tree-MTR and iSOUP-Tree-MTR bagging (with 10 trees) approaches.

The FIMT-DD method is capable of detecting changes in the concept and adapting to them. However, this study is oriented towards comparing the local and global tree-based approaches on equal grounds, thus the change detection and adaptation mechanisms in FIMT-DD have been disabled for this study.

5 Results

In this, section we present and discuss the results of our experiments and provide insights into several of the observed phenomena.



(a) EUNITE03 dataset (b) SCM1d dataset

Fig. 2: The evolution of $\overline{MAE}$ on the EUNITE03 and SCM1d datasets.

Bicycles dataset performance. In Fig. 1, we show the mean absolute errors for the three targets of the Bicycles dataset. While the local approach is better than the iSOUP-Tree-MTR method on the registered count target (Fig. 1a), it is outperformed by iSOUP-Tree-MTR on both of the casual count (Fig. 1b) and the total count (Fig. 1c) variables.

Arguably, registered users are much more consistent in their behaviour and their numbers are more independent of the other variables, so the benefit of using a global method for this variable is diminished. The number of casual users and, consequently, the number of the total users, on the other hand, are much less consistent, therefore, the availability of additional information from all of the targets greatly improves the performance in the global approach.

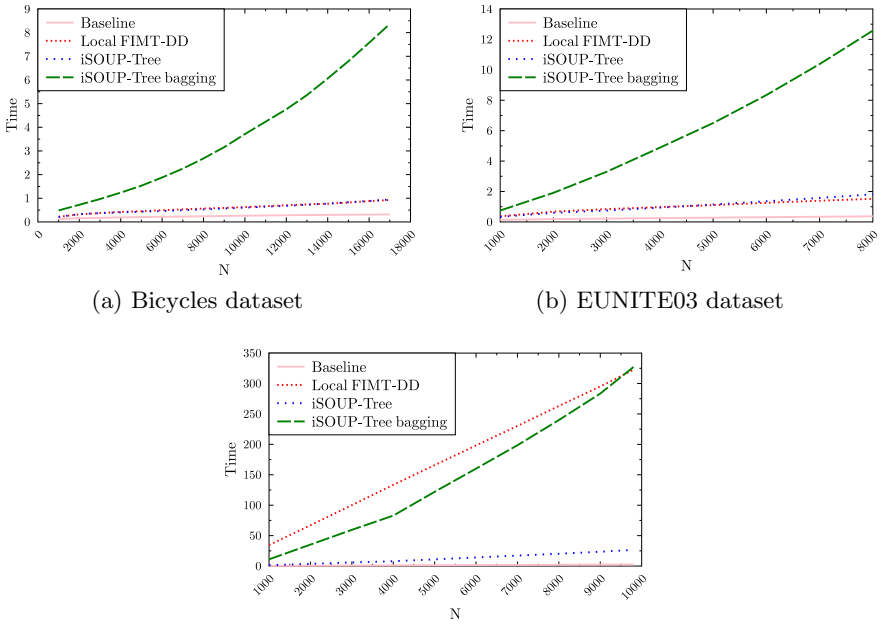
For all three targets, the iSOUP-Tree-MTR bagging method outperformed all of the other methods, including the basic iSOUP-Tree-MTR method.

EUNITE03 dataset performance. With regard to the first experimental question, the results on the EUNITE03 are a clear-cut triumph of the global iSOUP-Tree-MTR method, as its mean absolute error is consistently lower than that of the local FIMT-DD approach (see Fig. 2a). Notably, the baseline also has comparably good predictive performance. This possibly implies that the baseline has a bias towards the type of dependencies present in the EUNITE03 dataset.

In contrast to the Bicycles dataset, bagging performs worse than the base iSOUP-Tree method, and is, surprisingly, comparable to the baseline. However, these results (excluding the results above, from the local versus global discussion) are much less consistent than on the Bicycles dataset.

SCM1d dataset performance. At first glance, the results on the SCM1d dataset (see Fig. 2b) appear similar to those on the EUNITE03 dataset. The global iSOUP-Tree-MTR outperforms the local FIMT-DD method and the bagging iSOUP-Tree-MTR method outperforms the base iSOUP-Tree-MTR method.

However, none of the tree-based approaches beat the baseline. For global methods, this occurs due to the splitting mechanism of the iSOUP-Tree-MTR



(a) Bicycles dataset

(b) EUNITE03 dataset

(c) SCM1d dataset

Fig. 3: The time consumption of tree-based methods.

method. Since the SCM1d dataset has a high number of input attributes, many of them provide similar contribution to the heuristic, so the splitting mechanism cannot determine the best candidate among the attributes. As the examples accumulate, a tie threshold is reached and a split is made with lower confidence. In this case, apparently, an inappropriate split is selected and the performance suffers. For details on the tie breaking mechanism, see Ikonomovska et al. [7].

6 Conclusions and Further Work

Obviously, this does not affect the baseline, so its performance continues to improve. Additionally, this also does not affect the local FIMT-DD method, as discriminating candidate attributes is much easier when considering only one target. However, FIMT-DD still performs much worse than all other approaches.

Time consumption. The results from the time consumption (see Fig. 3) do not provide any surprises. When the number of targets is low, i.e., on the Bicycles and EUNITE03 datasets, the local FIMT-DD and global iSOUP-Tree-MTR approaches use comparable amounts of time, with iSOUP-Tree-MTR using slightly more. While the local method has to produce more trees, the global method has to use more complex data structures to encode and store information about the

multiple targets. Further experiments will however be needed to determine what are the differences in the time consumption when the number of targets grows, as the results in Fig. 3c are not representative for models of similar sizes, due to the splitting problem described above. The fact that the local FIMT-DD method uses approximately the same amount of time as the bagging method, which produces 10 trees, is highly indicative that, were it not for the splitting problem, iSOUP-Tree-MTR would still use considerably less time.

From our experiments, we can conclude that the global method iSOUP-Tree-MTR generally outperforms the local FIMT-DD method, or, at least, does not have a significantly worse performance. The iSOUP-Tree-MTR method is, however, sensitive to the situation in which a large number of attributes have a similar potential contribution, which severely stifles the growth of the trees.

While we intuitively expect that the global approach takes less time than the local approach, when the number of targets is high, we do not have the sufficiently extensive results required to confirm this. The splitting problem encountered on the SCM1d dataset distorts the results with respect to time, as the local models grow in complexity, while the global model practically did not grow at all. Further analysis on datasets where both approaches grow models is required.

Our results also warrant further exploration of the difference between the basic iSOUP-Tree-MTR method and it's bagging counterpart, as the results on the three datasets are inconclusive. It appears that bagging does offer some advantages over the basic method, however, it is unclear weather these will manifest themselves through increased predictive performance. Also, unlike the single-target study, we have, in two of the datasets, shown significant improvement when using bagging.

Overall, the global approach does have merit in the streaming context and can produce significant improvements in predictive performance for tree-based methods. These improvements are, however, reliant on the potential dependencies or independencies of the targets. The more independent the targets, the better we would expect the local method to perform, and vice versa, the more dependent the targets, the better the expected performance of the global method.

In our future work, we plan to perform an extended experimental evaluation on additional datasets. We will also consider additional performance evaluation measures, e.g., relative error, which would enable the observation of average errors even on datasets where the targets do not fall in the same range. Furthermore, we plan to evaluate the memory consumption of the compared methods. We also intend to implement option trees [8] within the global iSOUP-Tree-MTR approach: these have not only been show to grow faster with fewer available examples, but which consider multiple attributes for splits. This should help curtail the problem of many input attributes observed on the SCM1d dataset.

Acknowledgments

The authors are supported by The Slovenian Research Agency (Grant P2-0103 and a young researcher grant) and the European Commission (Grants ICT-2013-612944 MAESTRA and KT-2013-604102 HBP).

References

1. Appice, A., Džeroski, S.: Stepwise induction of multi-target model trees. In: 18th European Conference on Machine Learning. pp. 502–509 (2007)
2. Bifet, A., Gavaldà, R.: Adaptive learning from evolving data streams. In: 8th International Symposium on Advances in Intelligent Data Analysis VIII. pp. 249–260 (2009)
3. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive online analysis. The Journal of Machine Learning Research 11, 1601–1604 (2010)
4. Domingos, P., Hulten, G.: Mining high-speed data streams. In: 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 71–80. ACM, New York (2000)
5. Duarte, J., Gama, J.: Ensembles of adaptive model rules from high-speed data streams. In: 3rd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications. pp. 198–213 (2014)
6. Fanaee-T, H., Gama, J.: Event labeling combining ensemble detectors and background knowledge. Progress in Artificial Intelligence pp. 1–15 (2013)
7. Ikonomovska, E., Gama, J.: Learning model trees from data streams. In: 11th International Conference on Discovery Science. pp. 52–63 (2008)
8. Ikonomovska, E., Gama, J., Džeroski, S.: Online tree-based ensembles and option trees for regression on evolving data streams. Neurocomputing 150, 458–470 (2015)
9. Ikonomovska, E., Gama, J., Džeroski, S.: Incremental multi-target model trees for data streams. In: 2011 ACM Symposium on Applied Computing. pp. 988–993. ACM, New York (2011)
10. Ikonomovska, E., Gama, J., Džeroski, S.: Learning model trees from evolving data streams. Data Mining and Knowledge Discovery 23(1), 128–168 (2011)
11. Jr., C.N.S., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovvery 22(1-2), 31–72 (2011)
12. Kocev, D., Vens, C., Struyf, J., Džeroski, S.: Tree ensembles for predicting structured outputs. Pattern Recognition 46(3), 817–833 (2013)
13. Osojnik, A., Panov, P., Džeroski, S.: Multi-label classiffication via multi-target regression on data streams. In: 18th International Conference on Discovery Science (to appear) (2015)
14. Oza, N.C., Russel, S.J.: Experimental comparisons of online and batch versions of bagging and boosting. In: 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 359–364. ACM, New York (2001)
15. Rutkowski, L., Pietruczuk, L., Duda, P., Jaworski, M.: Decision trees for mining data streams based on the McDiarmid's bound. IEEE Transactions in Knowledge and Data Engineering 25(6), 1272–1279 (2013)
16. Shaker, A., Hüllermeier, E.: IBLStreams: a system for instance-based classification and regression on data streams. Evolving Systems 3(4), 235–249 (2012)
17. Struyf, J., Dzeroski, S.: Constraint based induction of multi-objective regression trees. In: 4th International Workshop on Knowledge Discovery in Inductive Databases. pp. 222–233 (2005)
18. Xioufis, E.S., Groves, W., Tsoumakas, G., Vlahavas, I.P.: Multi-label classification methods for multi-target regression. CoRR abs/1211.6581 (2012), `http://arxiv.org/abs/1211.6581`

A Hardware-Based Approach for Frequent Itemset Mining in Data Streams

Lázaro Bustio[1,2], Raudel Hernández[1], René Cumplido[2], Claudia Feregrino[2], and José M. Bande[1]

[1] Advanced Technologies Application Center.
7^a ♯ 21812 e/ 218 y 222, Rpto. Siboney, Playa, C.P. 12200, Havana, Cuba.
{lbustio,jbande,rhernandez}@cenatav.co.cu
[2] National Institute for Astrophysics, Optics and Electronic.
Luis Enrique Erro No 1, Sta. Ma. Tonantzintla, 72840, Puebla, México.
{rcumplido,cferegrino}@ccc.inaoep.mx

Abstract. Data streams are unbounded and infinite flows of data arriving at high rates which cannot be stored for offline processing. Because of this, classical approaches for Data Mining cannot be used straightforwardly in data stream scenario. This paper introduces the first reported single-pass hardware-based algorithm for frequent itemset mining on data streams. Experimental results of the hardware implementation of the proposed algorithm are also presented and discussed.

Keywords: Data Mining, Frequent Itemset Mining, Data streams, Reconfigurable Hardware, Parallel Algorithms

1 Introduction

Data Mining is a research area that investigates the tools and techniques needed to efficiently extract information from large volumes of data. One particularly important technique in Data Mining is Frequent Itemset Mining aimed at discovering those sets of items that can be found together more than a given number of occurrences in a data set(named *frequent itemset*) [1].

One scenario that is gaining attention is Data Streams Mining. Frequent itemset mining on data streams is incipient and the majority of the developed algorithms for this task cannot deal with data streams in an exhaustive fashion because of high incoming rates of data, short processing times needed, and the impossibility of storing the incoming stream. Data streams mining can be found in video and audio streams, network traffic, commercial transactions, among others [11], but such applications need to operate at high speed so hardware-based approaches have been proposed to achieve that goal. In such approaches, custom hardware architectures can be used as processing platforms due to their capacity to exploit parallelism.

In this paper, a parallel algorithm for frequent itemset mining on data streams (which was designed to be implemented and executed in hardware) that uses a Landmark Window Model [8] is proposed. This algorithm is based on a systolic

tree which is well suited for data streams handling. To prove the feasibility of this algorithm, several experiments were conducted showing that it outperforms some well known algorithms of the state-of-the-art selected as baseline.

This paper is structured as follows: in section 2 the theoretical basis that support this research is presented. A review of state-of-the-art is addressed in Section 3 while Section 4 introduces the proposed algorithm. Results are shown in Section 5 and finally, conclusions are given in Section 6.

2 Theoretical basis

Let $I = \{i_1, i_2, .., i_n\}$ be a set of n items and T be a transactional dataset[3]:

Definition 1 (Itemset). *An itemset X is a set of items over I such that $X \subseteq I$.*

Definition 2 (Transaction). *A transaction $t \in T$ over I is a couple $t = (tid, X)$ where tid is the transaction identifier, and $X \subseteq I$.*

Definition 3 (Support). *The support of an itemset X is the fraction of transactions in T containing X.*

An itemset is called *frequent* if its support is greater than a given minimal support threshold *minsup*.

Definition 4 (Data stream). *A data stream is a continuous, unbounded and not necessarily ordered, real-time sequence of data items.*

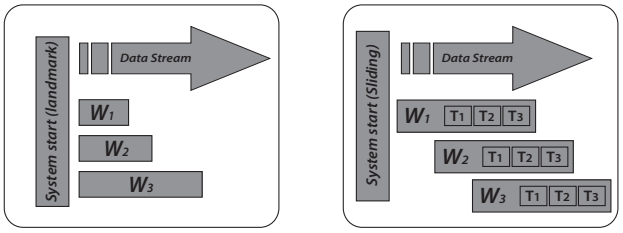
In data stream three main characteristics are presented[2, 11, 16]:

- *Continuity.* Items in stream arrive continuously at a high rate.
- *Expiration.* Items can be accessed and processed just once.
- *Infinity.* The total number of data is unbounded and potentially infinite.

Definition 5 (Window). *A window in a data stream is an excerpt of transactions.*

Windows can be constructed using one of following approaches [13, 8]:

- *Landmark window model.* This model employs some point (called landmark) to start recording where a window begins. The support count of an itemset i is the number of transactions containing i between the landmark and the current time (see Figure 1a).
- *Sliding window model.* This model uses the latest W transactions in the mining process. As newest transactions arrive, oldest in the sliding windows are excluded. This model can be compared with a FIFO queue. The use of this model imposes a restriction: as some transactions will be included in the mining process, methods for finding expired transactions and discounting the frequency counting of the itemsets involved are required (see Figure 1b).



(a) Landmark Window Model. (b) Sliding Window Model.

Fig. 1: Different windows model.

In this paper, Landmark Window Model was selected because it can be seen as a general case (and starting point) of others complex windows model such as Sliding. Also, there are applications which Landmark Window is better suited than others methods [18] (to detect buyers' behavioral patterns in a supermarket during a season, for instance).

2.1 Reconfigurable Computing

Reconfigurable Hardware Computing is referred to the use of hardware devices in which the functionality of the logic gates is customizable at run-time, and FPGAs are the main exponent of this approach. The architecture of FPGAs is based on a large number of logic blocks which perform basic logic functions. Because of this, FPGAs can implement from a simple logical gate, to a complex mathematical function. FPGAs can be reprogrammed; that is, the circuit can be "erased" and then, a new architecture that implements a brand new algorithm can be placed. This capability of the FPGAs allows the creation of fully customized architectures, reducing cost and technological risks that are present in traditional circuits designs [9].

3 Related works

A state-of-the-art of frequent itemsets mining in hardware can be organized as it is shown in Table 1. Analyzing the revised literature, it can be noticed that there are three preferred approaches: algorithms that use Apriori [1] as the starting point, algorithms that use FP-Growth [12] and those that use Eclat [24]. Algorithms that mimic the Apriori[3, 4, 23, 22] in hardware require loading the candidates itemsets and the dataset into the hardware. This strategy is limited by the capacity of the chosen platform: if the number of transactions to manage is larger than the hardware capacity, transactions must be loaded separately

[3] Dataset is referred to databases, unstructured data file, relational databases or any other data source. In this paper, dataset is used to refer data streams.

in many consecutive times degrading overall performance (the acceleration obtained by the use of FPGA is lost in communication). These issues are forbidden in data streams mining. One valid option for frequent itemsets mining on data streams is to develop a tree-based approaches where transactions flow inside the tree.

Table 1: Main architectures for Frequent Itemsets Mining.

Title	**Year**
Apriori-based	
Efficient Hardware Data Mining with the Apriori Algorithm on FPGAs.[3]	2005
An Architecture for Efficient Hardware Data Mining Using Reconfigurable Computing Systems.[4]	2006
Hardware-Enhanced Association Rules Mining With Hashing and Pipelining.[23]	2008
Novel Strategies for Hardware Acceleration of Frequent Itemset Mining With the Apriori Algorithm.[22]	2009
Eclat-based	
An FPGA-based Accelerator For Frequent Itemset Mining.[25]	2013
Accelerating intersection Computation In Frequent Itemset Mining With FPGAs.[17]	2013
FP-Growth-based	
Mining Association Rules with Systolic Trees.[20]	2008
A Reconfigurable Platform for Frequent Pattern Mining.[19]	2008
A Highly Parallel Algorithm for Frequent Itemset Mining.[15]	2010
Design and Analysis of a Reconfigurable Platform for Frequent Pattern Mining.[21]	2011

Similarly to Apriori-based algorithms, the FP-Growth-based algorithms [20, 19, 15, 21] need to copy the mining dataset to the processing platform. They also require two passes over the dataset, except Mesa et al. [15], however it still needs to download the dataset to the hardware device. This is impractical in data streams mining scenario due to the Expiration restriction. Like other reviewed algorithms, authors focused their attention in better data structures that allow the efficient counting of frequent itemsets. However, algorithms based on FP-Growth use the FP-Tree data structure [12] that is based on prefix-trees which are well suited for data streams mining applications.

Eclat-based algorithms [25, 17] use the vertical dataset representation in order to save memory and processing time. In [25, 17], authors use the intersection

of items to compute the support showing that it is more efficient than hash-trees. All the Eclat-based implementations propose an hybrid approach, where the most time and/or memory consuming functions were downloaded to custom hardware while the software controls the execution flow and data structures. Although the vertical dataset representation allows to save memory and processing time, it is not compatible with the Expiration restriction of data streams.

4 Proposed method

The basic idea of the proposed method is to develop a tree structure of processing units where transactions on data streams flow from the root node to the leaf nodes. The tree structure can be seen as a *systolic tree* where each of its nodes has one bottom node (child) and one right node.

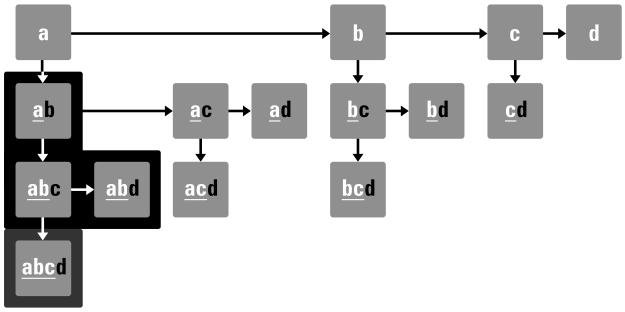


Fig. 2: Systolic tree data structure used in data stream mining. Remarked section shows a sub-tree with nodes that have the itemset ab of a transaction t as root.

Figure 2 shows the systolic tree where vertically-arranged nodes represent a prefix path and parent nodes contain the prefix itemset (that represent transactions of data streams) for their children. Taking a random node r, the sub-tree who had the node r as root is formed by all possible combinations of items with the itemset stored in r as their prefix, this leads to recursive mining strategies. The systolic tree data structure implements a distributed control scheme where processing and control logic are distributed among the nodes. A schematic of the systolic tree is shown in figure 3 and starting from it, the algorithm 1 is proposed.

Algorithm 1 is executed simultaneously in each node of the systolic tree. For each transaction t in Landmark Window when a new itemset X arrives to a node occupied by an item $i_j \in I$, one of following decisions must be taken:

1. If $\{i_j\} \subseteq X$ then the frequency counter of the node is incremented and the itemset $X - \{i_j\}$ is flowed to child and right node.

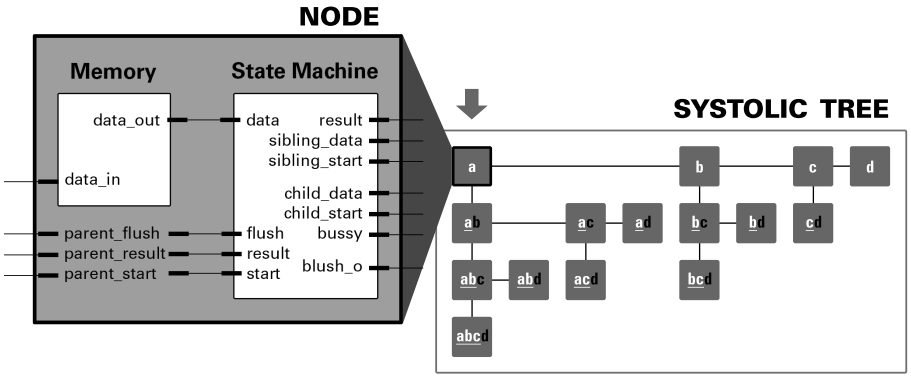


Fig. 3: Schematic of the processing node and structure of the systolic tree.

2. If $\{i_j\} \not\subseteq X$ then X is flowed to right node.

Algorithm 1 simultaneously uses Depth First Search (DFS) and Breadth First Search (BFS) traversals (from lines 20 to 25) allowing a high parallelism: two dimensional search is performed concurrently. These DFS and BFS traversal strategies are also employed to determine which itemsets can be regarded as frequent once the frequency counting of each itemset was computed and stored in nodes of the systolic tree.

After the frequency of each itemset is computed, a backtracking strategy using the Apriori [1] property which states that:

- All subsets of a frequent itemset are frequent.
- Any super-set of an infrequent itemset is also infrequent.

is employed to obtain the frequent itemsets from the systolic tree. This strategy is implemented in algorithm 2. Once again, a simultaneous BFS and DFS strategy is implemented lines 7 to 10.

Theoretically, the size (in number of nodes) of the systolic tree is determined by $|I|$, where I is a super-set of items in the incoming data stream (see section 2). Since $|I|$ cannot be established a priori (due the Infinity property of data streams), the size of the systolic tree will be determined by the capacity of the development platform. Assuming that the development platform contains enough computational resources (ideal case), the size of the systolic tree (in number of nodes) that can hold any streams formed by items of I will be:

$$nodes = 2^{|I|} - 1 \tag{1}$$

In a real case, the available resources of a FPGAs are limited. Suppose 100% of area occupation in selected hardware device, a number k will be the maximum number of processing nodes that can be supported by the selected device. Then

Algorithm 1: Parallel algorithm for finding the frequency counting of flushed transactions in a Landmark Window.

Input: Landmark Window *landmark_window*
Output: Systolic tree with the counting frequency of each itemset.

```
1  n_i ← systolyc_tree.RootNode;
2  foreach transaction t in landmark_window do
3      Traverse(t, n_i);
4  Procedure Traverse(t, n_i);
5  if n_i.IsOccupied == false then
6      n_i.IsOccupied = true;
7      n_i.Item = t.ItemAt(0);
8      FlushInParallel(t, n_i);
9  else
10     if t.Contain(n_i.Item) == true then
11         FlushInParallel(t, n_i);
12     else
13         ñ_i ← n_i.RightNode;
14         Traverse(t, ñ_i);
15 EndProcedure;
16 Procedure FlushInParallel(t, n_i);
17 n_i.Counter + +;
18 t̃ = t.Exclude(n_i.Item);
19 if t̃.IsEmpty == false then
20     StartParallalelBlock:;
21     ñ_i ⟵ n_i.ChildNode;
22     Traverse(t̃, ñ_i);
23     ñ_i ⟵ n_i.RightNode;
24     Traverse(t̃, ñ_i);
25     EndParallelBlock;;
26 EndProcedure;
27 return systolic_tree;
```

the maximum number max_{items} of items in I in the incoming data stream that can be handled by the chosen device will be:

$$max_{items} = \log_2(k+1) \tag{2}$$

For example, if the selected hardware device can hold 1024 nodes (k) in systolic tree, then the maximum number of items of I that can be handled will be 10 ($|I|$).

It is important to notice that for a certain number n of items in set I, if

$$2^n - 1 > k \tag{3}$$

Algorithm 2: Parallel algorithm for finding frequent itemsets.

Input: Flushing flag $flush$, Minimum support value min_sup.
Output: Frequent itemsets and their frequency counting $< itemset, frequency >$.

```
1  n_i ← systolyc_tree.RootNode;
2  if (flush == true) then
3      FlushMethod(n_i, min_sup);
4  Procedure FlushMethod(n_i, min_sup)
5  if (n_i.counter ≥ min_sup) then
6      if (n_i.ChildNode! = null)and(n_i.RightNode! = null) then
7          StartParallalelBlock:
8          FlushMethod(n_i.ChildNode, min_sup);
9          FlushMethod(n_i.RightNode, min_sup);
10         EndParallelBlock;
11         n_i.GatewayMode = true;
12     else
13         FlushResult.Add(n_i, n_i.Counter);
14 else
15     FlushResult.Add(n_i, n_i.Counter);
16 return FlushResult;
```

the systolic tree cannot hold all possible combinations of itemsets and therefore some itemsets will not be taken into account during the mining process. In other words, the number of processing nodes needed to handle all possible combinations of itemsets generated from I exceeds the maximum number of processing nodes that can be mapped into the selected FPGA. Here, mining will be approximate with no false positives produced. On the contrary, if

$$2^n - 1 \leq k \tag{4}$$

the systolic tree can hold all the possible combinations of itemsets, therefore the mining process will be exact. At this point, a conclusion arises: the proposed algorithm is exact in its theoretical basis, but in practice its accuracy is determined by the available area in the chosen development platform.

To obtain frequent items, a recursive strategy that uses the Apriori [1] property stated before is used. In this strategy, if a node is declared as *frequent*, then its child and right nodes must be processed recursively to determine whether they are frequent or not. On the contrary, if a node is regarded as *infrequent*, its descendants will be infrequent and process can be stopped (consequence of the Apriori property). This optimizes the traverse strategy to obtain the frequent itemsets in the systolic tree.

5 Results

The proposed method was modeled in VHDL using Xilinx ISE Suite 14.2 and targeted for Virtex 5 XC5VLX330T device. After the architecture was synthesized and implemented, the occupied area is 90.3 %, holding 1161216 processing nodes. This allows to handle a maximum of 20 different items, while the largest itemsets handled by FPGA-accelerated architecture reported is about 11 items. The maximum operation frequency obtained for this architecture is 238 Mhz. This means that the proposed architecture can process $5.07x10^6$ transactions (containing at most 20 items) per second (worst case), this derives in a throughput of 1.19 Gbps. Besides, as far as we know, this is the first FPGA-accelerated architecture for frequent itemset mining over data streams reported.

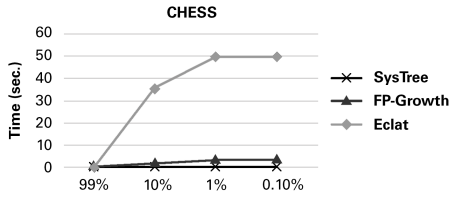
To validate the performance of the proposed systolic tree architecture, some experiments were conducted simulating data streams with 4 different datasets. MSNBC and Chess datasets were taken from UCI repository [14] and the other two were created with the Almaden IBM Synthetic Dataset Generator [10]. Employed datasets are described in Table 2.

Table 2: Dataset specifications.

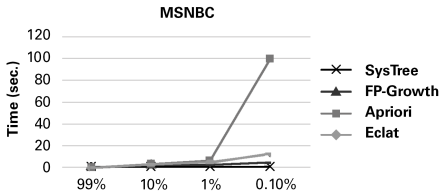
Dataset	**Size (Mb)**	**#Trans**	**#Items**	**Ave.IT**
MSNBC	4.219	989 818	17	2.82
Chess	0.340	3 196	75	37
T10I20D100K	1.468	100 000	20	10
T20I20D1M	27.123	1 000 000	20	20

Since there are no hardware architectures reported for frequent itemset mining on data streams, the best software implementations (concerning of performance and resources consumption) of the Apriori [5], FP-Growth [6] and Eclat [5] algorithms that were reported in the state-of-the-art, were used as baseline. These implementations were downloaded from Borgelt's website [7] and several experiments were conducted using MSNBC, Chess, T10I20D100K and T20I20D1M datasets on an Intel Core i3 at 1.8GHz and 4Gb of RAM. Experiments using all datasets were conducted in software using several *minsup* values and timing results were compared versus the same experiments conducted in hardware. Timing results are shown in fig. 4. These results are shown to give an idea of the performance of the proposed method compared against some well established state-of-art algorithms. Window size were set to 100 000 transactions. This allow to handle in one window the MSNBC, Chess and T10I10D100K datasets, while T20I20D1M were separated in 10 windows.

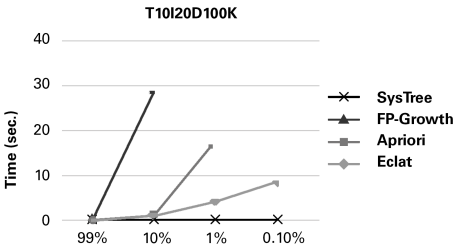
As it is shown in fig. 4, the proposed architecture outperforms all the baseline algorithms in all datasets. Note that, for the sake of clarity, processing times that exceeds 3 minutes are not shown. All datasets, except Chess, con-



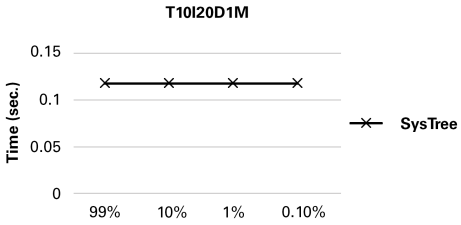
(a) Execution time for the proposed architecture with Chess dataset.



(b) Execution time for the proposed architecture with MSNBC dataset.



(c) Execution time for the proposed architecture with T20I20D100K synthetic dataset.



(d) Execution time for the proposed architecture with T10I20D1M synthetic dataset. Here, timing results that exceeds 300s were dropped from chart.

Fig. 4: Performance evaluation of the proposed algorithm and architecture for different datasets. X axis represent the variation of the support threshold expressed in %.

tain 20 or less single items, so they can be mapped directly into the hardware architecture. Chess dataset contains 75 items so the systolic tree architecture only takes into account for the mining process the first 20 items that arrive. In this case, the mining process is approximate, discovering a subset of total frequent itemsets. Itemsets regarded as frequent in this case have the same frequency counting as if they had been calculated with any baseline algorithm. For MSNBC, T10I20D100K and T20I20D1M, the mining process is exact and it was performed one order of magnitude times faster than baseline algorithms.

Experiments demonstrate that the proposed architecture is insensitive to variations in support threshold: this is explained because of all hardware needed for mining incoming data stream is available and it is the same whether the minimum support threshold is 1%or 99%. Restrictive support threshold conduce to less frequent itemsets while relaxed support threshold conduce to more frequent itemsets. Performance of software implementations of Apriori, Eclat and FP-Growth are quite sensitive in its performance to changes on minsup value.

6 Conclusions.

This paper introduces a new parallel algorithm for frequent itemset mining in data streams which is designed to be implemented in a custom hardware architecture. The proposed algorithm is based on a tree data structure that allows to increase the mining performance. The corresponding hardware architecture is based on a systolic tree approach where the control logic is distributed among all processing nodes. Experimental results showed that the hardware architecture is able to extract correctly all itemsets from data streams with a significant speed up when compared against software-based implementations.

References

1. Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.
2. Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 1–16, New York, NY, USA, 2002. ACM.
3. Zachary K. Baker and Viktor K. Prasanna. Efficient hardware data mining with the apriori algorithm on FPGAs. In *Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, FCCM '05, pages 3–12, Washington, DC, USA, 2005. IEEE Computer Society.
4. Zachary K. Baker and Viktor K. Prasanna. An architecture for efficient hardware data mining using reconfigurable computing systems. In *Field-Programmable Custom Computing Machines, 2006. FCCM '06. 14th Annual IEEE Symposium on*, pages 67–75, April 2006.
5. C. Borgelt. Efficient implementations of Apriori and Eclat. In *FIMI*, volume 90 of *CEUR Workshop Proc.* CEUR-WS.org, 2003.
6. C. Borgelt. An implementation of the FP-growth algorithm. In *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, OSDM '05, pages 1–5. ACM, 2005.
7. Christian Borgelt. Software for frequent pattern mining. `http://www.borgelt.net/fpm.html`. Accessed: 2015-05-20.
8. James Cheng, Yiping Ke, and Wilfred Ng. A survey on algorithms for mining frequent itemsets over data streams. *Knowl. Inf. Syst.*, 16(1):1–27, 2008.
9. Katherine Compton and Scott Hauck. Reconfigurable computing: A survey of systems and software. *ACM Comput. Surv.*, 34(2):171–210, June 2002.
10. IBM Corporation. IBM quest market-basket synthetic data generator. `http://www.cs.loyola.edu/~cgiannel/assoc_gen.html`. Accessed: 2015-04-20.
11. L. Golab and M. Ozsu. Data stream management issues - a survey. *Sigmod Record*, 2003.
12. Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 1–12, New York, NY, USA, 2000. ACM.

13. Ruoming Jin and Gagan Agrawal. Frequent pattern mining in data streams. In CharuC. Aggarwal, editor, *Data Streams*, volume 31 of *Advances in Database Systems*, pages 61–84. Springer US, 2007.
14. M. Lichman. UCI machine learning repository, 2013. Accessed: 2015-04-20.
15. Alejandro Mesa, Claudia Feregrino-Uribe, Rene Cumplido, and Jose Hernandez-Palancar. A highly parallel algorithm for frequent itemset mining. In Jose Francisco Martinez-Trinidad, Jesus Ariel Carrasco-Ochoa, and Josef Kittler, editors, *Advances in Pattern Recognition*, volume 6256 of *Lecture Notes in Computer Science*, pages 291–300. Springer Berlin Heidelberg, 2010.
16. Mr Pramod S. and O. P Vyas. Data stream mining: A review on windowing approach. *Global Journal of Computer Science and Technology*, 12(11-C), July 2012.
17. Shaobo Shi, Yue Qi, and Qin Wang. Accelerating intersection computation in frequent itemset mining with FPGA. In *High Performance Computing and Communications 2013, 2013 IEEE 10th International Conference on*, pages 659–665, Nov 2013.
18. Bai-En Shie, Philip S. Yu, and Vincent S. Tseng. Efficient algorithms for mining maximal high utility itemsets from data streams with different models. *Expert Systems with Applications*, 39(17):12947 – 12960, 2012.
19. Song Sun, M. Steffen, and J. Zambreno. A reconfigurable platform for frequent pattern mining. In *Reconfigurable Computing and FPGAs, 2008. ReConFig '08. International Conference on*, pages 55–60, Dec 2008.
20. Song Sun and J. Zambreno. Mining association rules with systolic trees. In *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, pages 143–148, Sept 2008.
21. Song Sun and J. Zambreno. Design and analysis of a reconfigurable platform for frequent pattern mining. *Parallel and Distributed Systems, IEEE Transactions on*, 22(9):1497–1505, Sept 2011.
22. D.W. Thoni and A. Strey. Novel strategies for hardware acceleration of frequent itemset mining with the apriori algorithm. In *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, pages 489–492, Aug 2009.
23. Ying-Hsiang Wen, Jen-Wei Huang, and Ming-Syan Chen. Hardware-enhanced association rule mining with hashing and pipelining. *Knowledge and Data Engineering, IEEE Transactions on*, 20(6):784–795, June 2008.
24. M. J. Zaki. Scalable algorithms for association mining. *IEEE Trans. on Knowl. and Data Eng.*, 12(3):372–390, May 2000.
25. Yan Zhang, Fan Zhang, Zheming Jin, and Jason D. Bakos. An FPGA-based accelerator for frequent itemset mining. *ACM Trans. Reconfigurable Technol. Syst.*, 6(1):2:1–2:17, May 2013.

Discovering and Tracking Organizational Structures in Event Logs

Annalisa Appice, Marco Di Pietro, Claudio Greco, and Donato Malerba

Dipartimento di Informatica, Università degli Studi di Bari Aldo Moro
via Orabona, 4 - 70126 Bari - Italy
{annalisa.appice, donato.malerba}@uniba.it

Abstract. The goal of process mining is to extract process-related information by observing events recorded in event logs. An event is an activity initiated or completed by a resource at a certain time point. Organizational mining is a subfield of process mining that focuses on the organizational perspective of a business process. It considers the resource attribute and derives a profile that characterizes the behavior of a resource in a specific business process. By relating resources associated with correlated profiles, it is possible to define a social network. This paper focuses on the idea of performing organizational mining of event logs via social network mining. It presents a framework that resorts to a stream representation of an event log. It adapts the time-based window model to process this stream, so that window-based social resource networks can be constructed, in order to represent interactions between resources operating at the data window level. Finally, it integrates specific algorithms, in order to discover (overlapping) communities of resources and track the evolution of these communities over consecutive windows. This paper applies the defined framework to a real event log.

1 Introduction

Event logs are data sets currently produced by several information systems (e.g. workflow management systems). They contain the executions (called traces) of a business process. A trace is defined as an ordered list of activities invoked by a resource from the beginning of its execution to the end. Process mining refers to the discovery, conformance and enhancement of process models from event logs. By tightly coupling event logs and process models, process mining makes possible to detect deviations, predict delays, support decision making and recommend process redesigns.

Thus far, the majority of the focus of process mining research has been on control flow, i.e. the ordering of activities. However, as discussed in [12], process mining can go beyond the control flow perspective. In particular, the organizational perspective can be considered by focusing on the resources, i.e. which performers are involved in the process model and the way they are related.

Organizational mining focuses on the organization perspective by learning more about people, machines, organizational roles, work distribution and work

patterns [10]. Song and van der Aalst [10], seminally, introduced the social network analysis as a comprehensive approach towards organizational mining. The nodes in a social resource network correspond to organizational entities which are, in general, one-to-one associated with resources. The arcs in a social resource network correspond to relationships between such organizational entities. Various ways have been developed in [13], in order to construct social resource networks from event logs. Quantifying the similarity of two resources is just one of many ways of constructing a social network. Every resource can be associated with a profile, i.e. a vector that describes the relevant features of a resource, while the distance between two profiles can be quantified by using well-known distance measures. A few studies [14, 13] investigated how social networks analysis can be tailored for organizational mining. They mainly base on converting an event log into a social network of resources and generating social network metrics, in order to determine relevant organizational patterns. For example, between (a ratio based on the number of geodesic paths visiting a given node) can be used to find possible bottlenecks.

On the other hand, the discovery of organizational patterns cannot neglect that one of the most relevant features of social networks is the community structure of the network [8], i.e. the organization of nodes in communities, with many arcs joining nodes of the same cluster and comparatively few arcs joining nodes of different communities. Such communities can be considered as fairly independent compartments (namely organizational structure) of a network, playing a similar role. Based upon the idea that identifying communities in a network technically is finding node clusters in graphs, Song and van der Aalst [10] applied various clustering algorithms, in order to discover similar resources grouped in the same organizational structure. However, network data pose specific challenges to classical clustering algorithms [5] that we cannot neglect when looking for organizational patterns in process mining. Traditional clustering works on the distance or similarity matrix, but network data structure leads to specific algorithms using the graph property directly (k-clique, quasi-clique, vertex-betweenness, edge-betweeness etc).

Several community detection algorithms, exploiting graph information, have been investigated in social network analysis (see [8, 4] for recent surveys). However, to the best of our knowledge, performances of these algorithms are still unexplored in the organization perspective of the process mining scenario. A further limit of the seminal research of Song and van der Aalst [10] is that they used clustering to detect communities of resources in an event log, which was processed as a static, finite dataset. However, this static view can be restrictive, as it neglects the real case of an event log that is continuously fed with new events generated from (new) running traces. On the other hand, a dynamic event log is expected to feed a dynamic social resource network, which may change over time (new resources are active, old resources are inactive). This poses the process mining problem of discovering and tracking changes in resource communities (organization structures) of a business process. This problem is also consistent with a recent trend of research [11, 3, 7] in social network analysis, which has started

to consider the problem of tracking the progress of communities over time in a dynamic social network.

In this paper, we formalize an event-based model of the log that is handled as a stream of events. We address the task of discovering and tracking time evolving resource communities in process mining. The stream is produced by several running traces of a specific business process. Every event in the stream is time stamped, belongs to one running trace, is performed by a specific resource and executes a certain activity. A time-based window model is used to decompose the stream into consecutive windows. We formulate a two-stepped stream learning framework, named TOSTracker (Time-evolving Organizational Structure Tracker), in order to perform organizational mining of streamed resources of a business process. In the on-line phase, event data produced by running traces are queried, in order to extract a social resource network, while a community detection algorithm (i.e. Louvain algorithm [1]) is applied, in order to determine communities covering specific organizational roles. It is noteworthy that, in order to detect "overlapping" communities, we investigate the idea representing the arcs of the social resource network as nodes of a linear network and apply Louvain algorithm to this linear network. In the off-line phase, the evolution (e.g. birth, death, merge, split, contraction and expansion) of discovered time-evolving communities is tracked over time.

The paper is organized as follows. In Section 2, we report basic concepts of this study, while in Section 3 we describe the organizational mining framework. In Section 4, we analyze the performances of this framework in a case study with real data. Finally, conclusions are drawn and future works are sketched.

2 Basics

The premise is that an *event log* $\mathcal{L}$ is a set of events concerning a certain business process type $\mathcal{P}$. An *event* $\epsilon(tid, a, r, t)$ is characterized by a set of mandatory characteristics, that is, the event corresponds to an activity a, is triggered by a resource r and has a timestamp t that represents date and time of occurrence. In addition, each event in the log is linked to a particular trace tid and is globally unique. A *trace* $\mathcal{T}$ represents the execution of a business process instance. It is a finite sequence of distinct events, that is,

$$\mathcal{T} = \epsilon(tid, a_1, r_1, t_1), \epsilon(tid, a_2, r_2, t_2), \ldots, \epsilon(tid, a_n, r_n, t_n), \tag{1}$$

such that all events of $\epsilon(tid, a_i, r_i, t_i) \in \mathcal{T}$ are linked to a specific trace tid, while time is non-decreasing in the trace (i.e. for $1 \leq i < j \leq n$: $t_i \leq t_j$). According to this definition of trace, an event log is traditionally dealt as a static bag of full traces of a specific business process [12]. An example showing a fragment of an event log, organized around the concept of trace, is reported in Table 1.

In this paper, we move from a static perspective to a dynamic perspective when handling event logs. We consider an event log as a dynamic dataset that is continuously fed with new events generated by traces, which are still running. This allows us to handle an event log according to an *event stream* model. In

Table 1. A fragment of an example event log. Each event is linked to a specific trace. It corresponds to an activity, has a timestamp and is triggered by a resource.

TraceId	Activity	Timestamp	Resource
1	Register request (R)	2010-12-30:11:02	Pete
1	Examine throughly (ET)	2010-12-31:10:06	Sue
1	Check ticket (CT)	2011-01-05:15:12	Mike
1	Decide (D)	2011-01-06:11:18	Sara
1	Reject request (RR)	2011-01-07:14:24	Pete
2	Register request (R)	2010-12-30:11:32	Mike
2	Check ticket (CT)	2010-12-30:12:12	Mike
2	Examine causally (EC)	2010-12-30:14:16	Pete
2	Check ticket (CT)	2010-12-31:15:31	Pete
2	Decide (D)	2011-01-05:11:22	Sara
2	Pay compensation (PC)	2011-01-08:12:05	Ellen
3	Register request (R)	2010-12-30:14:32	Pete
3	...	...	...
...	...	...	...

particular, an event stream $\mathcal{S}$ is an ordered, unbounded sequence of time stamped events:

$$\mathcal{S} = \epsilon(tid_1, a_1, r_1, t_1), \epsilon(tid_2, a_2, r_2, t_2), \ldots, \epsilon(tid_i, a_i, r_i, t_i), \ldots \tag{2}$$

where events of the stream arrive sequentially, at consecutive time points (i.e. $t_i \leq t_{i+1}$), from the bag of running traces of business process P. The bag of running traces may change over time, since old executions can be completed, while new executions can be started at a certain time point. The event stream associated with the fragment of event log reported in Table 1 is shown in Table 2.

An event stream, like any data stream, is unbounded in length. It is impractical to query all the data of a stream. Windows are commonly used stream approaches to query open-ended data. Instead of computing an answer over the whole data stream, the query (or operator) is computed, maybe several times, over a finite subset of events. Several window models are defined in the literature. In this study, we consider the *time-based window model* [2], which decomposes an event stream into consecutive (non overlapping) windows of fixed temporal size. When a window is completed, it is queried. The answer is stored in a data synopsis for the mining phase, while the windowed data are discarded. Formally, let $\Delta(T)$ be the window temporal size of the model, a time-based window model decomposes a stream $\mathcal{S}$ into non overlapping windows,

$$\mathcal{S}(\Delta(T)) = t \rightarrow t+\Delta(T), t+\Delta(T) \rightarrow t+2\Delta(T), \ldots, t+(j-1)\Delta(T) \rightarrow t+j\Delta(T), \ldots \tag{3}$$

so that every window $t + (j-1)\Delta(T) \rightarrow t + j\Delta(T)$ selects stream events $\epsilon(tid_i, a_i, t_i, r_i) \in \mathcal{S}$ acquired at each time point $t \in [t+(j-1)\Delta(T), t+j\Delta(T)[$.

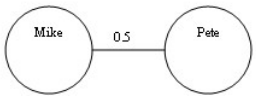
As a query operator, we consider a *social resource network* constructor. Consequently, as a data synopsis for the data storage, we use a *weighted graph*. The nodes of this social network (or equivalently graph data synopsis) correspond to resources triggering one or more events in the data window. Each resource is associated with a *resource-activity profile* that represents the number of times a

Table 2. The stream model of the event log reported in Table 1. The event stream is divided into consecutive windows, which are 24 hours long according to the time-based window model with $\Delta(t) = 24h$.

TraceId	Activity	**Timestamp**	Resource
1	Register request (R)	2010-12-30:11:02	Pete
2	Register request (R)	2010-12-30:11:32	Mike
2	Check ticket (CT)	2010-12-30:12:12	Mike
2	Examine causally (EC)	2010-12-30:14:16	Pete
3	Register request (R)	2010-12-30:14:32	Pete
2	Check ticket (CT)	2010-12-31:15:31	Pete
1	Examine throughly (ET)	2010-12-31:10:06	Sue
2	Decide (D)	2011-01-05:11:22	Sara
1	Check ticket (CT)	2011-01-05:15:12	Mike
1	Decide (D)	2011-01-06:11:18	Sara
1	Reject request (RR)	2011-01-07:14:24	Pete
2	Pay compensation (PC)	2011-01-08:12:05	Ellen
...	...	...	...

Fig. 1. The activity resource profile of Pete and Mike (left side). Both are resources active in the data window [2010-12-30:0:00, 2010-12-30:24:00] of the stream reported in Table 2. The social resource network extracted from this data window (right side) contains two nodes associated to resources Pete and Mike, as well as an arc connecting these resources. The weight associated to the arc is the Pearson correlation coefficient computed between the activity resource profiles of the edged nodes.

Resource	Register Request	Check ticket	Examine Causally
Mike	1	1	0
Pete	2	1	1



resource performs an activity in the data window. Arcs between nodes are associated with weights that express the importance of the relations. As in [10], for each pair of resources r_i and r_j, we compute the Pearson correlation coefficient of the resource-activity profiles, which are associated with r_i and r_j, respectively. Formally, $w(r_i, r_j) = \frac{\sum_A r_i(A) r_j(A) - n\overline{r_i}\overline{r_j}}{\sqrt{\sum_A r_i(A)^2 - n\overline{r_i}^2}\sqrt{\sum_A r_j(A)^2 - n\overline{r_j}^2}}$ where A denotes an activity associated with the resource profiles, $r_i(A)$ $(r_j(A))$ is the number of times A is performed by r_i (r_j) in the data window, $\overline{r_i(A)}$ $(\overline{r_j(A)})$ is the average $\overline{r_i(A)} = \frac{1}{n}\sum_A r_i(A)$ $(\overline{r_j(A)} = \frac{1}{n}\sum_A r_j(A))$ and n is the number of activities in the resource profile. We rank potential arcs according to the Pearson correlation values associated with. Arcs associated with the top p ranked Pearson correlation are, finally, added to the graph data synopsis. An example of a social resource network is reported in Figure 1. Alternative metrics, which can be computed to estimate the weight of an arc, can monitor handover of work or subcontracting [10].

3 Time-evolving Organization Structure Tracker

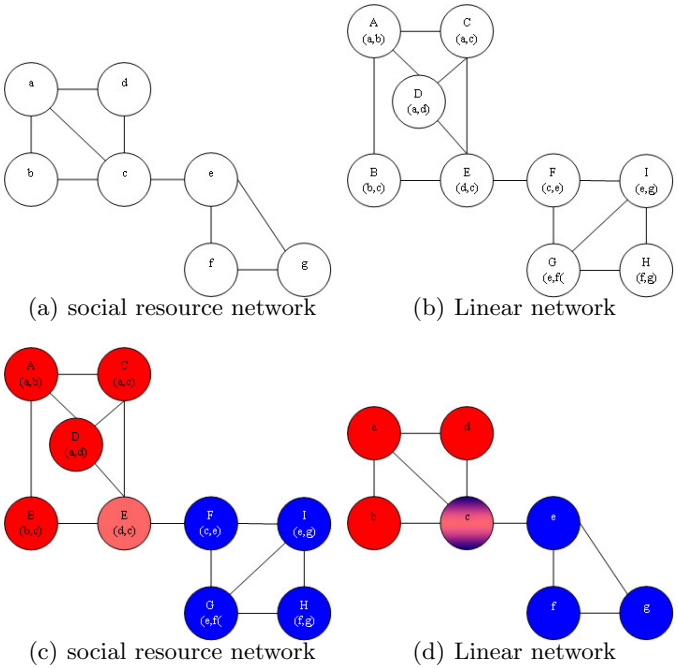
The organizational mining framework, called TOSTracker, operates in two phases. The on-line phase consumes events as they arrive from the event stream and analyzes the buffered events, window-by-window, in order to determine a social resource network, which is stored in a graph data synopsis. For each window, (overlapping) communities are detected from the social resource network stored in the graph-based data synopsis (see details in Section 2). The set of resource communities is then discovered from this data synopsis as a model of the organization structure of the business process along the time horizon associated to the processed event window. Since this resource community model is stored in a database, while windowed events are definitely discarded, resource communities represent the organizational knowledge for the future off-line query phase. The off-line phase, which is repeatable, tracks the evolution of resource communities discovered along a query time horizon and retrieved from the database.

3.1 Resource community detection

Resource community detection is performed by resorting to Louvain algorithm [1]. This is a greedy optimization that attempts to optimize the "modularity" of a partition of the network.

The modularity is a measure of the structure of a network. It is designed to measure the strength of division of a network into communities (or clusters). Formally, modularity is the fraction of the arcs that fall within the given communities minus the expected such fraction if edges were distributed at random. So, for a given division of the network's nodes into some communities, modularity reflects the concentration of arcs within modules compared with random distribution of arcs between all nodes regardless of communities. Networks with high modularity have dense connections between the nodes within communities, but sparse connections between nodes in different communities. In this study, the randomization of the arcs is done according to the Configuration model presented in [6], as it allows us to generate a randomization preserving the node degrees of the original network. The modularity is computed according to the measure of Reichardt-Bornholdt [9].

Let $\mathcal{G}(\mathcal{N}, \mathcal{A})$ be a network, N be the set of nodes and A be the set of arcs. The optimization of modularity is performed in two steps. In the first step, Louvain algorithm looks for "small" communities by optimizing modularity locally. So it, initially, assigns a different community to each node $u \in \mathcal{N}$. Hence, in this initial partition, there are as many communities as there are nodes. Then, for each node $u \in \mathcal{N}$, Louvain algorithm considers neighbors v of u (i.e. each v is edged to u in $\mathcal{A}$) and evaluates the gain of modularity that would take place in the network by transferring u from its community to the the community of v. The node u is, finally, transferred to the community for which this gain is positive and maximum. If no positive gain is possible, it stays in its original community. This process is applied repeatedly and sequentially for all nodes until no further improvement can be achieved and the first phase is then complete.



(a) social resource network (b) Linear network

(c) social resource network (d) Linear network

Fig. 2. Overlapping resource community discovery: the social resource network (see Figure 2(a)) is transformed into the linear network (see Figure 2(b)). Disjoint communities discovered by Louvain algorithm in the linear network (see Figure 2(c)) are mapped into overlapping communities of resources (see Figure 2(d)). We note that node c of the social resource network belongs to the red community with degree 0.75 and to the blue community with the degree 0.25.

In the second step, Louvain algorithm aggregates nodes belonging to the same community and builds a new network whose nodes are the communities. This step of the algorithm consists in building a new network whose nodes are now the communities found during the first phase. The weights of the arc edging two community nodes is given by the sum of the weight of the arcs between nodes in the corresponding two communities. Once this new network is computed, the first step of the algorithm is applied to the resulting weighted network. In this way, the two steps are repeated iteratively until a maximum of modularity is attained and a hierarchy of communities is produced.

We note that Louvain algorithm is an efficient and easy-to-implement algorithm for identifying communities in networks. However, it allows us to detect only "disjoint" communities. In contrast, organizational structures of a business process may be overlapping in a social resource network (i.e. a resource may place two roles in the same business process). In order to discover overlapping resource communities, we have applied Louvain algorithm to the linear network that is constructed from the social resource network. This linear network is con-

structed by representing the arcs of the social resource network as nodes of the linear network. Let u and v be two nodes of the linear network, so that u denotes the arc (u_i, u_j, w_u) of the social resource network, while v denotes the arc (v_i, v_j, w_v) of the social resource network. There is an arc (u, v, w) in the linear network iff arcs (u_i, u_j, w_u) and (v_i, v_j, w_v) share one vertex in the social resource network (i.e. $u_i = v_i$ or $u_i = v_j$ or $u_j = v_i$ or $u_j = v_j$). Let x be the vertex shared between (u_i, u_j, w_u) and (v_i, v_j, w_v), weight $w = w_u/(deg(x) - w_v)$ where the $deg(x)$ is the sum of weights associated to arcs incoming/outcoming x in the social resource network (see Figures 2(a)-2(b)).

Disjoint communities discovered in the linear network are then mapped into possibly overlapping communities discovered in the social resource network. A node u_i of the social resource network belongs to every community discovered in the linear network, which groups at least one node u of the linear network, so that u is associated it an arc of the social resource network with a vertex in u_i (see Figures 2(c)-2(d)). Let u_i a node of the social resource network, the degree according to the resource u_i belongs to a community C can be computed as follows:

$$degree(u_i, C) = \frac{|\{C' \in C_{u_i} | C' = C\}|}{|\{C_{u_i}\}|} \tag{4}$$

where C_{u_i} is the set of communities assigned to arcs of I_{u_i} in the linear network, while I_{u_i} is the set of arcs of the social resource network incoming/outcoming u_i (see Figure 2(d)).

3.2 Tracking evolutions of resource communities

A dynamic resource community is represented as a time line, that is, a sequence of (evolving) resource communities, ordered by time, with at least one resource community for each time point t. The evolutions of dynamic resource communities along their time line is expressed in terms of birth, death, merge, split, expansion and contraction [3] (see Figure 3). Intuitively, the *birth* event describes a new resource community observed at time t with no corresponding resource community in the set of communities already tracked. A new community is created with time line starting at the birth time. The *death* event describes the dissolution of a resource community that does not appear for several consecutive time points. The timeline of this community ends when it disappears. The *merge* event occurs when two or more distinct resource communities observed at time $t-1$ can be similar to a single community at time t. A branch is added to connect the single communities at time $t-1$ to the merged community created at time t . The single communities, subsequently, share the same time line. The *split* event occurs when a single resource community present at time $t-1$ can be similar to two or more distinct resource communities at time t. A branching occurs from the starting community to the split ones with the creation of an additional resource community that shares the timeline of up to time $t-1$, but has a distinct timeline from time t onwards. The *expansion* of a resource community occurs when its cardinality grows at time t. The *contraction* of a resource community occurs when its cardinality decreases at time t. The Jaccard coefficient

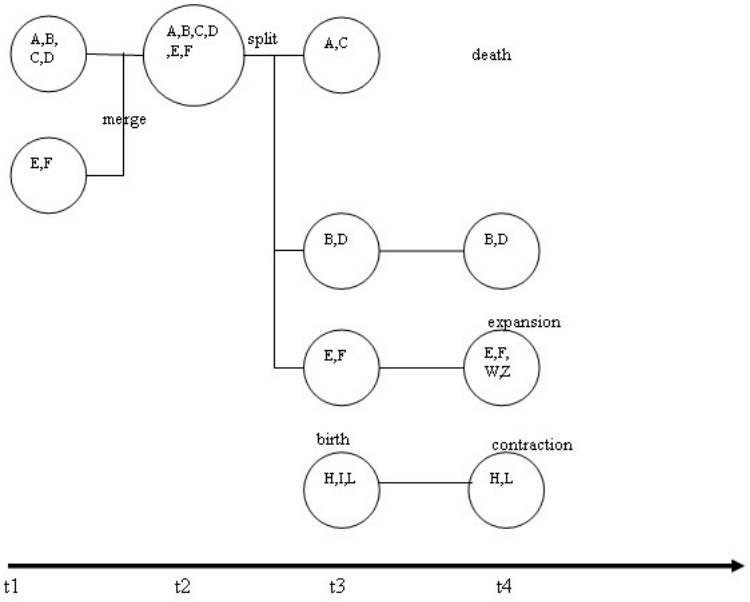


Fig. 3. The time line of the evolutions of dynamic resource communities

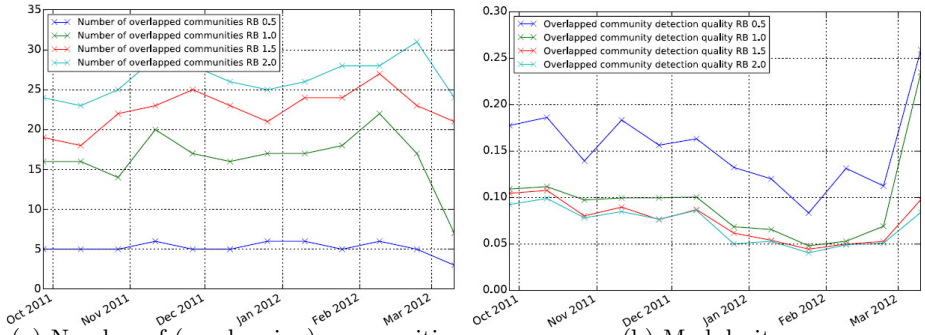
can be computed, in order to estimate the similarity between communities and detect these events.

The evolution is tracked according to the algorithm described in [3]. This algorithm is independent on the algorithm selected, in order to discover online the resource communities. It is based on the analysis of the degree of similarity between pairs of communities discovered at consecutive time points. The input is a time series $\mathcal{T}$ of resource community sets (i.e. $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_t, \ldots, \mathcal{C}_n$), where each set $\mathcal{C}_t$ represents the organizational structure (i.e. set of resource communities) extracted over a specific time horizon and time stamped with t in $\mathcal{T}$.

The tracking algorithm is two stepped. In the initialization phase, the algorithm constructs a dynamic resource community for each resource community $C \in \mathcal{C}_1$. The same community is also added to the set of frontier communities $\mathcal{F}$. In the mining step, the algorithm iteratively analyzes each organization structure $\mathcal{C}_t$ (with $t = 2, 3, n$) by computing the similarity between communities of $\mathcal{C}_t$ and frontier communities of $\mathcal{F}$.

Formally, for every $C \in \mathcal{C}_t$ and $C_{\mathcal{F}} \in \mathcal{F}$, $jaccard(C, C_{\mathcal{F}})$ is computed with $jaccard(\cdot, \cdot)$ the Jaccard coefficient. The compared communities are similar if and only if their Jaccard coefficient exceeds a user-defined threshold σ. We note that the output of the similarity computation may reveal series of community evolution events. These events are detected, in order to update accordingly the time lines of the dynamic resource communities, as well as the set of frontier communities. Let us consider $C \in \mathcal{C}_t$. If there is no frontier resource community $C_{\mathcal{F}} \in \mathcal{F}$ so that $jaccard(C, C_{\mathcal{F}}) \geq \sigma$ then a birth event is happened. A

new dynamic community containing C is created. If there is one and only one frontier resource community $C_{\mathcal{F}} \in \mathcal{F}$ so that $jaccard(C, C_{\mathcal{F}}) \geq \sigma$, this indicates the continuation of C. If there are two or more frontier resource community $C_{\mathcal{F}_1}, \ldots, C_{\mathcal{F}_k} \in \mathcal{F}$ so that $jaccard(C, C_{\mathcal{F}_i}) \geq \sigma$ $(i = 1, 2, \ldots, k)$, then a merge event is happened. The new merged community is connected to the time lines of the frontier communities contributing to the merge. On the other hand, let us consider $C_{\mathcal{F}} \in \mathcal{F}$. If there are two or more resource community $C_1, \ldots, C_k \in \mathcal{C}_t$ so that $jaccard(C_i, C_{\mathcal{F}}) \geq \sigma$ $(i = 1, 2, \ldots, k)$, then a split event is happened. Every new split community is connected to the time line of the frontier community that has originated the split. Finally let us consider the pair $(C, C_{\mathcal{F}}) \in \mathcal{C}_t \times \mathcal{F}$ so that $jaccard(C, C_{\mathcal{F}}) \geq \sigma$ then if $cardinality(C - C_{\mathcal{F}}) > 0$, then an expansion event is happened; if $cardinality(C_{\mathcal{F}} - C) > 0$, then a contraction event is manifested.



(a) Number of (overlapping) communities (b) Modularity

Fig. 4. Organizational structure discovery by varying the resolution parameter RB of Reichardt-Bornholdt measure between 0.5, 1, 1.5 and 2.

4 Case study

We have evaluated performances of the proposed framework by considering real traces of a business process taken from a Dutch Financial Institute and provided in the Business Processing Intelligence Challenge 2012. [1] The log contains 262.200 events from September 27, 2011 to March 10, 2012, 69 resources, in 13.087 traces. The business process is an application process for a personal loan or overdraft within a global financing organization. The event stream has been processed with time-based window model with $\Delta(T) = 15$ days. The resource social network of every window is buffered into a graph data synopsis, where

[1] http://www.win.tue.nl/bpi/2012/challenge

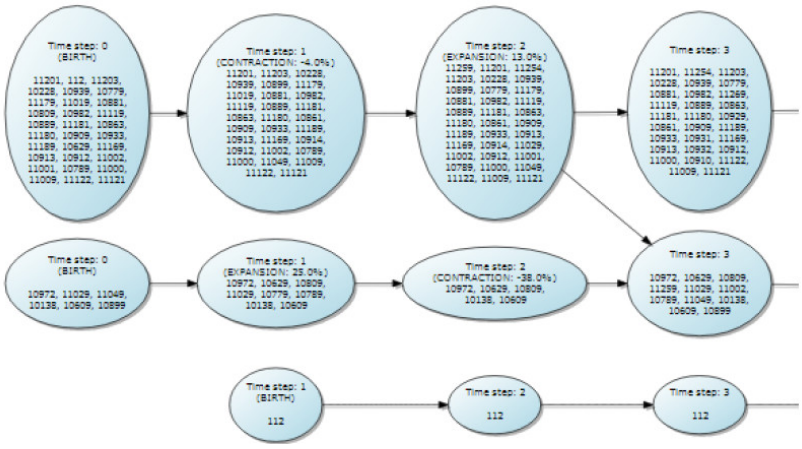


Fig. 5. A fragment of the time line of the time-evolving organizational structures.

nodes are associated with resources triggering an event in the window, arcs between nodes are materialized if the weight associated with the arc is in the top $p = 75\%$ Pearson correlation coefficients computed between acivity-resource profiles of nodes (see details in Section 2). The overlapping resource communities of the organizational structure are discovered in the data synopsis where event data are buffered, according to the algorithm described in Section 3.1. The evolution of the time-evolving organizational structure of a process is tracked according to the algorithm described in Section 3.2. As Lovain algorithm is run with Reichardt-Bornholdt measure [9] that requires a resolution parameter RB, we perform the community discovery by varying RB between 0.5, 1, 1.5 and 2. The number of discovered (overlapping) communities of resources is shown in Figure 4(a), while the modularity of the discovered organization structure is shown in Figure 4(b). We note that the highest modularity is achieved when the lowest number of communities is detected per window in correspondence of $RB = 0.5$. Finally, we track the evolution of the organizational structure discovered with $RB = 0.5$. An example of evolutions tracked in the discovered time-evolving organization structure is shown in Figure 5.

5 Conclusions

In this paper, we have described a framework to perform organizational mining of streamed resources of a business process. The stream is processed according to a time-based window model. Events batched in a window are queried, in order to extract a social resource network. A community detection algorithm is applied, in order to determine (overlapping) communities of resources covering specific organizational roles. The evolution (e.g. birth, death, merge, split, contraction and expansion) of the discovered communities is tracked over time. In this study,

social resource network is represented as undirected graph. As future work, we plan to extend this framework to process directed social resource networks.

6 Acknowledgments

This work fulfills the research objectives of the PON 02_00563_3470993 project "VINCENTE - A Virtual collective INtelligenCe ENvironment to develop sustainable Technology Entrepreneurship ecosystems", funded by the Italian Ministry of University and Research (MIUR), as well as the project "LOGIN- LOGistica INTegrata 2012-2015 (PII INDUSTRY 2015), announcement New Technologies for the Made in Italy.

References

1. V. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 10(P10008), 2008.
2. M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: a review. *ACM SIGMOD Record*, 34(2):18–26, 2005.
3. D. Greene, D. Doyle, and P. Cunningham. Tracking the evolution of communities in dynamic social networks. In *ASONAM 2010*, pages 176–183, 2010.
4. S. Harenberg, G. Bello, L. Gjeltema, S. Ranshous, J. Harlalka, R. Seay, K. Padmanabhan, and N. Samatova. Community detection in large-scale networks: a survey and empirical evaluation. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6):426–439, 2014.
5. T. Lei and L. Huan. *Community Detection and Mining in Social Media*. Morgan and Claypool Publishers, 2010.
6. M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036–104, 2006.
7. M. D. B. Oliveira, A. Guerreiro, and J. Gama. Dynamic communities in evolving customer networks: an analysis using landmark and sliding windows. *Social Netw. Analys. Mining*, 4(1):208, 2014.
8. M. Plantie and M. Crampes. Survey on social community detection. In *Social Media Retrieval*, Computer Communications and Networks, pages 65–85. Springer London, 2013.
9. J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Physical review E*, 74(1):016110, 2006.
10. M. Song and W. M. P. van der Aalst. Towards comprehensive support for organizational mining. *Decision Support Systems*, 46(1):300–317, 2008.
11. M. Spiliopoulou. Evolution in social networks: A survey. In *Social Network Data Analytics*, pages 149–175. Springer US, 2011.
12. W. M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.
13. W. M. P. van der Aalst, H. A. Reijers, and M. Song. Discovering social networks from event logs. *Computer Supported Cooperative Work*, 14(6):549–593, 2005.
14. W. M. P. van der Aalst and M. Song. Mining social networks: Uncovering interaction patterns in business processes. In *BPM 2004*, volume 3080 of *LNCS*, pages 244–260. Springer, 2004.

Intelligent Adaptation of Ensemble Size in Data Streams Using Online Learning

M. Kehinde Olorunnimbe, Herna L. Viktor and Eric Paquet

School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, ON, K1N 6N5, Canada
{molor068,hviktor}@uottawa.ca;eric.paquet@nrc-cnrc.gc.ca

Abstract. Online ensemble methods have been very successful to create accurate models against data streams that are susceptible to concept drift. The success of data stream mining has allowed diverse users to analyse their data in multiple domains, ranging from monitoring stock markets to analysing network traffic and exploring ATM transactions. Increasingly, data stream mining applications are running on mobile devices, utilizing the variety of data generated by sensors and network technologies. Subsequently, there has been a surge in interest in mobile (or so-called pocket) data stream mining, aiming to construct near real-time models. However, it follows that the computational resources are limited and that there is a need to adapt analytics to map the resource usage requirements. In this context, the resultant models produced by such algorithms should thus not only be highly accurate and be able to swiftly adapt to changes. Rather, the data mining techniques should also be fast, scalable, and efficient in terms of resource allocation. It then becomes important to consider Return on Investment (ROI) issues such as storage space needs and memory utilization. This paper introduces the Adaptive Ensemble Size (AES) algorithm, an extension of the Online Bagging method, to address this issue. Our AES method dynamically adapts the sizes of ensembles, based on the most recent memory usage requirements. Our results when comparing our AES algorithm with the state-of-the-art indicate that we are able to obtain a high Return on Investment (ROI) without compromising on the accuracy of the results.

Keywords: data streams, metalearning, adaptive ensemble size, return on investment, OzaBag

1 Introduction

In this era of the Internet of Things and Big Data, a proliferation of connected devices continuously produce massive amounts of fast evolving streaming data. A number of online learning methods have been highly successful in constructing accurate models against massive data streams. Success stories include retail store sales streamlining, chemical plant shutdown time prediction, and stock market monitoring, amongst others [8, 17]. Currently, the development of techniques to

facilitate mobile (or pocket) data stream mining is an emerging area of research with applications in the areas of business, telemedicine and security, amongst others. In this setting, the resultant models produced by data stream mining algorithms should not only be highly accurate and be able to swiftly adapt to changes. Rather, the learning techniques should also be efficient in terms of resource allocation. It then becomes important to consider issues such as storage space needs and memory utilization. This is especially relevant when we aim to build personalized, near-instant models for Big Data on small devices [8, 10].

This research addresses this emerging need for accurate, yet efficient, model construction [1, 15]. Our aim is to take an adaptive approach to resource allocation during the mining process. Consideration is given to the memory available to the algorithm and the speed at which data is processed. To this end, we introduce the Adaptive Ensemble Size (AES) technique that extends the Online Bagging (OzaBag) online ensemble learning algorithm. Our AES method takes advantage of the memory utilization cost, in order to vary the ensemble size during the data mining process. We aim to minimize the memory usage, while maintaining highly accurate models with a high utility. The reasoning behind our approach is based on the following observation. Intuitively, a higher change in memory utilization during stream classification potentially indicates a shift in the data at that particular stream window, which may occur as a result of concept drift. In this case, our AES algorithm increases the ensemble size in an attempt to maintain higher accuracies. The inverse is also true, in that we may be able to reduce the ensemble size, with no or little cost in terms of accuracies, when the properties of the data remain stable.

This paper is organized as follows. Section 2 introduces background work. In Section 3, we detail the Adaptive Ensemble Size (AES) algorithm. Section 4 discusses our experimental evaluation and results. Finally, Section 5 presents our conclusion and highlights future work.

2 Background

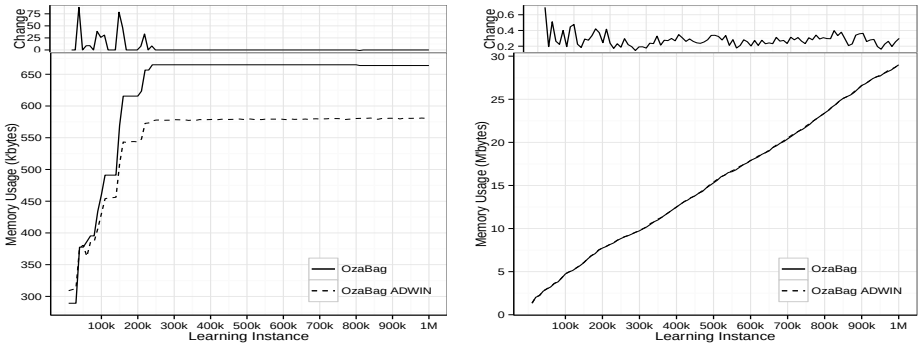
Online learners are well suited for data streams because of their ability to process chunks of data sequentially and to produce models from partial datasets. In online learning, an algorithm learns one instance at a time. This characteristic makes it an ideal learner for a streaming environment in which the instances arrive one after the other. Oza and Russell's Online Bootstrap Aggregating (OzaBag) is an online version of the well-known Bagging ensemble learning algorithm [5,13,14]. For a dataset N, while the Bagging algorithm assumes a binomial distribution because of the finite size of the dataset in memory, OzaBag assumes a Poisson distribution because of the continuous nature ($N \rightarrow \infty$) of the stream. Similar to Bagging algorithm, the most predicted class amongst the models is assigned to a new instance of the classification problem. OzaBag with ADWIN was presented as an extension of the OzaBag algorithm, in order to facilitate concept drift by increasing the rate of resampling [2]. Thus, ADWIN is in essence a change detector and estimator that keeps a variable-length window of recently

seen items. The maximal length of the window must be statistically consistent with the following hypothesis: there has been no change in the average value inside the window [7]. The two variants of the OzaBag algorithm is available in the Massive On-line Analysis (MOA) data stream mining framework [4]. In our research, we used the Hoeffding Tree method as base learner [7].

We studied the cost of building and applying data mining models while taking the memory usage of the ensemble, as well as the ensemble size, into consideration. To this end, we performed a series of preliminary experiments in order to observe the effects of concept drift on memory utilization. We also investigated the influence of ensemble sizes on the accuracy.

2.1 Size of Ensemble Object in Memory

In our work, we start from the observation that the size of an ensemble object in memory tends to be proportional to changes in characteristics of the learning instances. In this section, we present some preliminary experimentation in order to further ascertain this claim. To this end, we simulated two artificial data streams with MOA, using the LED data stream generator [4]. In our experiment, one stream is simulated with concept drift while the other is not. We conducted further experiments where we induced, and removed, concept drift at regular intervals as shown in Figure 2.



(a) Stream without concept drift (b) Stream with concept drift

Fig. 1: Memory Increase in the Data Stream

Figure 1, shows the memory usage and memory change as more instances are evaluated. The graphs indicate the memory usage while the learning process is being completed. As we can observe from Figure 1a, in a steady stream with no randomness or drift, the memory usage rapidly stagnates and becomes stable, as every new ensemble presents exactly the same size than the previous one. On the other hand, in Figure 1b, there is an increase in memory consumption in a stream

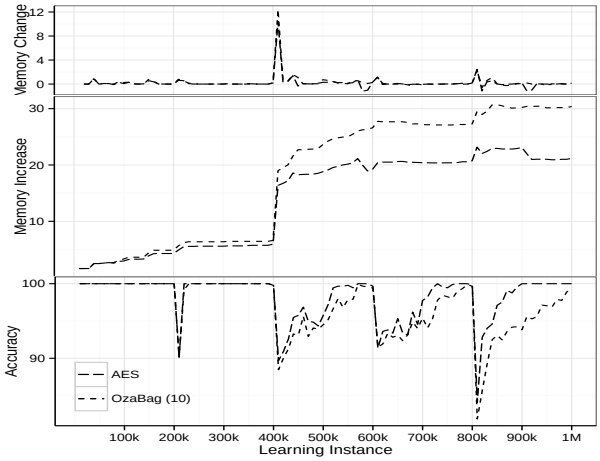


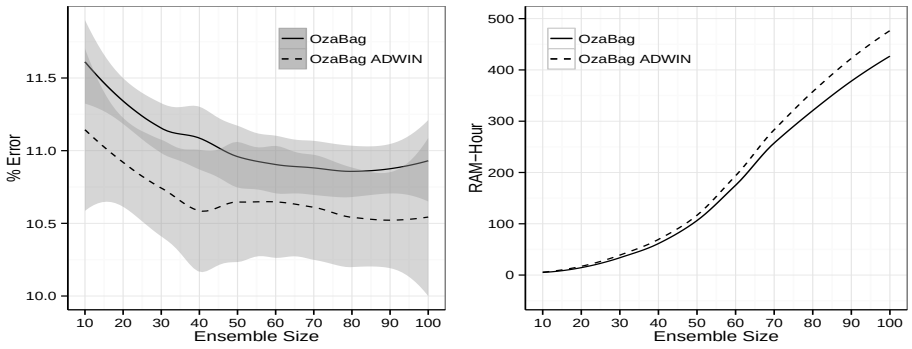
Fig. 2: Memory Change with Concept Drift induced at regular intervals

with concept drift. Also, generally speaking, both OzaBag and OzaBagADWIN have a similar memory usage, for this dataset. Finally, Figure 2 shows that there is a relationship between the induction of concept drift and memory utilization.

2.2 Ensemble Size versus Utility

Recall that our goal is to find a trade-off between accuracy and resource allocation, in a data stream setting. That is, our aim is follow a 'waste not, want not' approach to online learning. To that end, our algorithm is designed to maintain high accuracies while utilizing only the amount of resources that is required. The question of finding a trade-off between ensemble size and accuracy has been studied in [9] and [12]. It follows that the most appropriate answer is domain dependent and is influenced by numerous factors. Following [9] and [12], we performed experiments against the KDD CUP 1999 dataset, which is based on the DARPA98 network capture. The original raw training dataset represents 7 weeks of network traffic, containing 4,898,431 connection records. The test dataset comprises 2 weeks of data, corresponding to 311,029 connection records. Both contains 42 features, including the label. We used the OzaBag and OzaBagADWIN algorithms, varying the ensemble sizes from 10 to 100, by intervals of 10, with Hoeffding Trees as base learners. Figure 3a, shows the error (100% − accuracy) plotted against the ensemble size. As shown in the figure, the error rates are comparable against all ensemble sizes. That is, there is little gain in employing more base learners against this dataset. Figure 3b further shows that the computational cost of the algorithms dramatically increases when the ensemble size exceeds 20.

From Table 1, we further confirm that the computational cost of the algorithms considerably increases when the ensemble size exceeds 20. For instance, if we consider the lower cost OzaBag algorithm as an example, we notice from the



(a) Error with increase in ensemble size (b) RAM-Hour with increase in ensemble size

Fig. 3: Error and RAM-Hour With Increase in Ensemble Size

Table 1: Error, Time and RAM-Hour for Different Ensemble Sizes

	OzaBag Classifier			OzaBagADWIN Classifier		
Size	% Error	Time	RAM-Hr	% Error	Time	RAM-Hr
10	11.6899	258.1505	3.7057	11.1555	289.2415	4.0624
20	11.1423	545.5511	15.8006	10.7695	630.9148	19.4642
30	11.2444	828.5681	35.5382	10.7632	931.4664	39.7808
40	11.1055	1093.4734	62.5377	10.5776	1229.6467	70.7882
50	10.8791	1420.5763	102.2533	10.5187	1588.2306	114.2099
60	10.9803	1920.8247	175.1958	10.9398	2154.3582	188.7208
70	10.8465	2474.9559	252.9820	10.3325	2755.5237	288.2580
80	10.9129	3098.9755	354.4388	10.7108	3414.0819	375.2574
90	10.7657	2675.1987	348.0295	10.3227	3033.8762	413.6293
100	10.9787	3006.8569	435.9619	10.6252	3333.2110	477.4898

⋆ Time is measured in CPU Seconds

table that a variation of the ensemble size from 10 to 20 yields a small accuracy (k^+) improvement of 0.5474% but at the expense of a four folds increase in terms of cost. When the ensemble size is increased to 40, the accuracy increases by a meager 0.0364% but at the expense of a four folds increase in terms of cost as compared to an ensemble of size 20 and a seventeen folds increase as opposed to an ensemble size of 10. Based on these observations, we conclude that increasing the number of base learners often do not benefit the learning process, while it has a detrimental effect on the allocation of computational resources.

3 Adaptive Ensemble Size Algorithm

This section details or AES algorithm, as depicted in Algorithm 1. Our method adapts the size of the ensemble, depending on the memory cost of the current

window. By varying the ensemble size, our aim is to maintain the accuracy of the learning algorithm while guaranteeing high ROI [16].

Algorithm 1 AES Algorithm

Input: N is an evolving data stream ($N \rightarrow \infty$);
T is the training set drawn from N with examples x;
Y is the finite set of target class values y;
HT is the base learning algorithm (Hoeffding trees);
M is the number of models in the ensemble, with examples drawn from T;
min and max ensemble size values (set to 10 and 25, respectively);
δ is the generalized Kronecker function: $\delta(a,b) := \begin{cases} 1 \text{ if } a = b \\ 0 \text{ if } a \neq b \end{cases}$

initialize $A = 0$
for all T drawn from N **do**
  for all $m = 1$ to M **do**
    S_m = random sample of size d drawn from T, with replacement
    h_m = model induced by HT from S_m
    Compute average size of ensemble (bytes) $\rightarrow B$
    Set $k = Poisson(1)$
    for all $n = 1, 2, ..., k$ **do**
      Update h_m with current examples $S_m \in T$
  if $(B > A)$ & $(M < max)$ **then**
    $M \leftarrow M + 1$
  if $(B < A)$ & $(M > min)$ **then**
    $M \leftarrow M - 1$
  $A \leftarrow B$

Output: hypothesis: $h_{fin}(x) = argmax_{y \in Y} \sum_{m=1}^{M} \delta(y, h_m(x))$

The AES algorithm begins by drawing a training window, T, from a stream of evolving data, N. A total of M models (h_m) are induced independently from T. That is, we utilize an ensemble of models, h_m, from a number of random sample, $S_m \in T$, where m range from 1 to M. Recall that we use Hoeffding Trees as base learners. For a new classification instance, the predicted class with the highest number of votes is assigned to the new instance. The process samples bootstrap replicates from the data in the stream, using the Poisson probability distribution, and the models are continuously updated for each T.

At each classification instance, two variables, A and B, are maintained in order to records the average size (in bytes) of the ensemble in memory. These two parameters (A and B) are the previous and the current memory sizes of the ensemble respectively, with A initially set to zero. After each update of T, a total of M models (h_m) are induced, and the average memory size of h_m is stored in B. For every iteration of the algorithm, the values of A and B are compared. The size of the ensemble, M, is increased by 1 if $B > A$, since this implies that more memory is being utilized in the current window, compared to the previous window. On the other hand, the size of M is decreased by 1 if the value of B

is smaller than the value of A, since it indicates that the memory utilization is lower in the current window. This process continues as the data stream evolves.

We also implemented a variant of the AES algorithm called AES-ADWIN. The main differences between AES and AES-ADWIN is that the rate of resampling of AES-ADWIN is increased and that the windows A and B are of variable-length. That is, the AES algorithm is modified so that it utilizes the ADWIN change detection algorithm [3]. Subsequently, if ADWIN detects change in the error rate of one of the models h_i then we replace the model with the highest error with a new model.

In summary, our AES and AES-ADWIN methods are based on the observation that the increase in average memory usage points towards a potential change in the data distribution, which may occur due to some form of concept drift. When this happens, we increase the ensemble size in a bid to be pro-active and to obtain a better classification result. The converse is also true. That is, a decrease in memory usage of the ensemble indicates less difficulty in model construction, so the ensemble size may potentially be reduced. The size of M is kept between the allocated maximum and minimum values, i.e. $min \leq M \leq max$. By varying the size of M as we have done in the AES algorithm, we are able to conserve resources, without having to compromise on the accuracy of the classification results, as will be shown in the next section.

4 Experimentation

We used six real life datasets from the UCI[1] Machine Learning Repository in our experimentation, namely KDD, IMDb, Forest Cover Type, Electricity and Airline. All of these datasets are subject to concept drift.

Table 2: Summary of Datasets Used

Name	# of Records	# of Attributes	Characteristics
KDD'99 (10%)	494,021	42	Numeric, Nominal
Poker Hand	829,201	11	Numeric, Nominal
IMDb	120,919	1002	Numeric
Forest Covertype	581,012	55	Numeric, Nominal
Electricity	45,312	9	Numeric, Nominal
Airline	539,383	8	Numeric, Nominal

All experiments were performed using MOA [4], with the prequential evaluation (or the so-called 'test then train') method. The Kappa Statistics (k^+) was used as an accuracy measure since it has been shown to be particularly well adapted to streams with concept drift [4]. Recall that we are interested in measuring the Return on Investment (ROI) of the learning process. Recently, [16] introduced a ROI measure, where the emphasis is on comparing adaptation

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

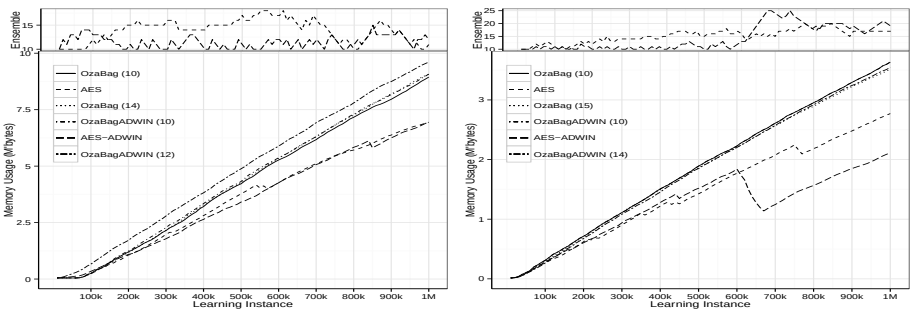
strategies, rather than contrasting ensemble approaches. We adapted this measure to derive the mean ROI of the evaluation steps. This allows us to make direct comparison between adapting and non-adapting algorithms and to measure the ROI in between steps of processed instances:

$$\overline{ROI} = \frac{1}{\mathcal{T} \times N} \sum_{i=1}^{\mathcal{T}} N \frac{\gamma_i}{\psi_i} = \frac{1}{\mathcal{T}} \sum_{i=1}^{\mathcal{T}} \frac{\gamma_i}{\psi_i} \tag{1}$$

where γ_i is the change in prediction accuracy (k^+); ψ_i is the cost (RAM-Hr); and $\mathcal{T}$ is the total number of windows.

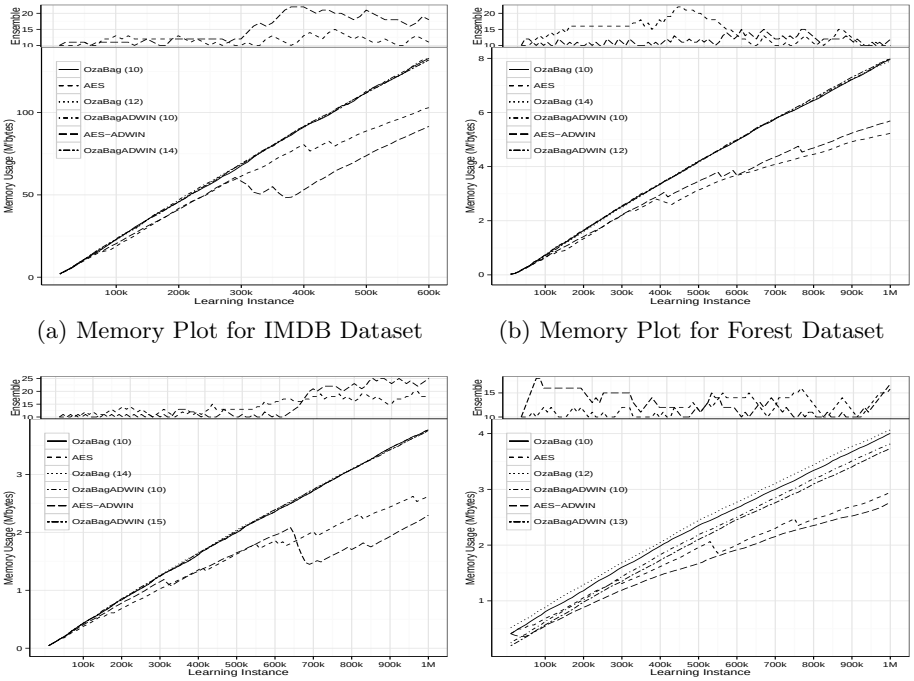
4.1 Results

In this section, we present our experimental results against the six datasets introduced in the previous section. Firstly, we turn our attention to memory utilization, as shown in Figures 4 and 5. The top parts of the subfigures show how the sizes of the AES and AES-ADWIN ensemble grow and shrink throughout the learning process. The bottom sections indicate the memory usage of the six classifiers we are employing, namely AES and AES-ADWIN (indicated in red and blue), as well as two variations of OzaBag and OzaBagADWIN. The first variation (Ozabag (10) and OzabagADWIN (10)) refer the default MOA settings, while the second variation refers to the ensembles with sizes equal to the number of base learners used by the AES and AES-ADWIN methods. Figures 4 and 5 clearly depicts that the sizes of the AES and AES-ADWIN ensembles vary considerably throughout the learning process, as based on the memory utilization within each window. The figures further show that the AES and AES-ADWIN algorithms compare favourably, in terms of memory utilization, to their OzaBag counterparts. This is the case even when the ensemble sizes increase to higher than the average values.



(a) Memory Plot for KDD Dataset (b) Memory Plot for Poker Dataset

Fig. 4: Change in Ensemble Size and Memory Utilization (A)



(a) Memory Plot for IMDB Dataset

(b) Memory Plot for Forest Dataset

(c) Memory Plot for Electricity Dataset

(d) Memory Plot for Airline Dataset

Fig. 5: Change in Ensemble Size and Memory Utilization (B)

We further compare the methods in terms of accuracy and ROI values. In these results, we show the average ensemble sizes we obtained, rounded to the nearest whole number. We also present the values for OzaBag and OzaBagADWIN for these average ensemble sizes. (Recall that OzaBag and OzaBagADWIN contains a fixed, static number of base learners.) The default numbers of base learners (10) for OzaBag and OzaBagADWIN also appear. A comparison of the k^+ accuracy results using the algorithms on the different datasets is shown in Table 3.

As clearly shown by the table, all four algorithms present highly similar levels of accuracy. The situation is quite different when the ROI is taken into account. Indeed, as shown by Table 4, our AES algorithm presents by far the best results in terms of ROI for all six (6) real life datasets. Our AES-ADWIN algorithm appears in second place.

Finally, Figures 6 and 7 shows a graphical representation of the ROI at each step of the mining process. One first notices that, for each dataset, there is initially a surge of the ROI. Such a behavior can be traced back to the initialization of the evaluation process, when the accuracy is building up. Recall that we are using the prequential evaluation method. This indicates that there is a rapid

Table 3: Kappa Plus Statistics Median for the datasets

	KDD'99			Poker			IMDb			Forest			Electricity			Airline		
	e.s	k^+	r	e.s	k^+	r	e.s	k^+	r	e.s	k^+	r	e.s	k^+	r	e.s	k^+	r
OzaBag	10	88.0362	6	10	88.7212	4	10	87.8519	5	10	88.3543	6	10	88.4178	2	10	88.2179	4
AES	14	88.6391	2	15	89.0466	2	12	88.4434	3	14	88.8211	3	14	88.2528	4	12	88.2114	5
OzaBag	14	88.5790	3	15	88.9772	3	12	**88.8997**	1	14	**89.0140**	1	14	88.1120	5	12	88.2061	6
OzaBagADWIN	10	88.5542	4	10	88.7212	4	10	88.3559	4	10	88.3910	5	10	88.4178	2	10	88.3647	3
AES-ADWIN	12	88.0736	5	14	88.4064	5	14	87.7557	6	12	88.4043	4	15	88.3007	3	13	88.3848	2
OzaBagADWIN	12	**89.1392**	1	14	**89.2080**	1	14	88.4804	2	12	88.8891	2	15	**88.9806**	1	13	**88.4034**	1

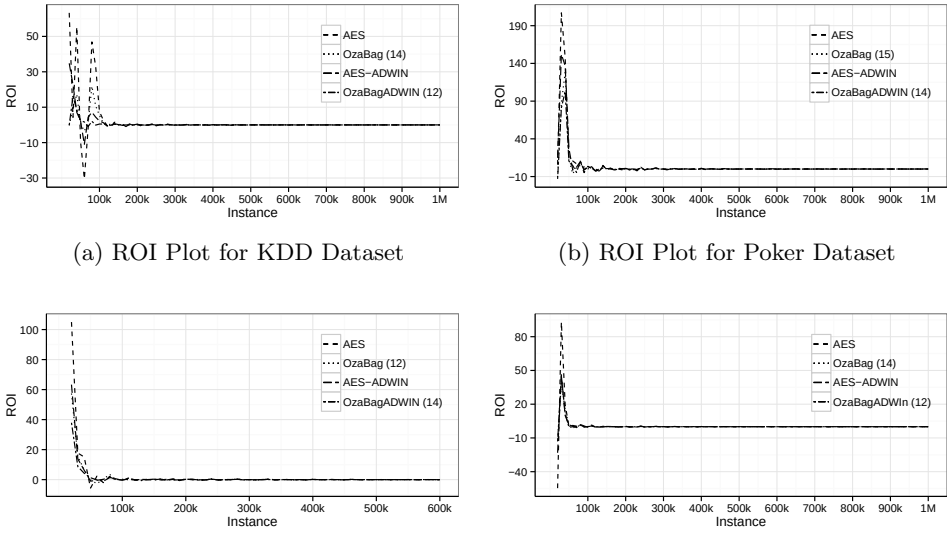
⋆ e.s = ensemble size; r = ranking (lower is better)

Table 4: ROI Comparison between the Algorithms

	KDD'99			Poker			IMDb			Forest			Electricity			Airline		
	e.s	ROI	r	e.s	ROI	r	e.s	ROI	r	e.s	ROI	r	e.s	ROI	r	e.s	ROI	r
AES	14	**1.7207**	1	15	**4.0195**	1	12	**1.3558**	1	14	**0.6184**	1	14	**1.2899**	1	12	**0.3871**	1
OzaBag	14	0.6557	3	15	2.2341	3	12	0.7991	3	14	0.4436	3	14	0.6790	3	12	0.1988	4
AES-ADWIN	12	0.7526	2	14	3.4481	2	14	0.8425	2	12	0.3732	4	15	1.1231	2	13	0.2344	3
OzaBagADWIN	12	0.3103	4	14	1.9285	4	14	0.5285	4	12	0.4511	2	15	0.5813	4	13	0.2477	2

⋆ e.s = ensemble size; r = ranking (lower is better)

buildup in accuracy at this point, before the latter peaks. Then, the ROI rapidly decreases, and remains essentially constant.



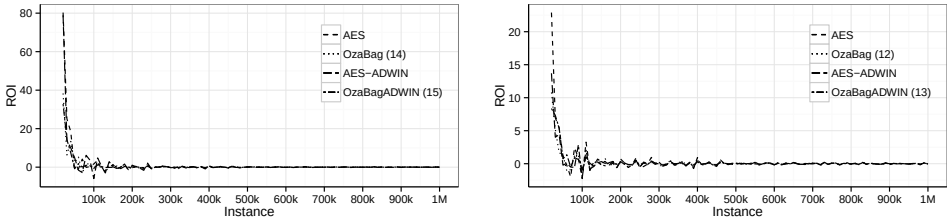
(a) ROI Plot for KDD Dataset

(b) ROI Plot for Poker Dataset

(c) ROI Plot for IMDb Dataset

(d) ROI Plot for Forest Dataset

Fig. 6: ROI Plots for KDD, Poker, IMDb and Forest Datasets



(a) ROI Plot for Electricity Dataset

(b) ROI Plot for Airline Datasets

Fig. 7: ROI Plots for Electricity and Airline Datasets

In summary, our results indicate that varying the ensemble size during online learning is beneficial, in that the amount of resources employed does not have a detrimental effect on the error rates. That is, the AES approach yields comparable results, in terms of accuracies, while the overall resource utilization, measured in terms of model construction time and memory utilization, is lowered. We believe that this initial result is worth exploring further. We are especially interested in extending our work to the area of mobile data mining, where there is a need for real-time data analytics on small devices that are susceptible to varying degrees and types of concept drift [6].

5 Conclusion

In a data stream setting, where we are potentially dealing with massive, fast evolving data, it is important to consider the overall utility of the learning process. This is especially relevant in a scenario where the resources are limited, for instance mobile data mining applications. This paper presented an online ensemble-based approach that adapts the size of an Online Bagging ensemble during learning from data streams. This property allows for flexibility in terms of resource usage and yields a high Return on Investment.

While our initial results are promising, a number of research avenues remain. Specifically, our next step would be to extend our work to specific case studies within the mobile data mining scenario. We are interested in exploring how the AES algorithm would perform when the types and degrees of concept drifts vary. More intelligent update rules, exploring different parameters will also be explored.

In this research, we did not discuss other factors that may also affect cost-sensitive adaptation, such as data labeling costs and economic utility considerations. We would also be interested in utilizing other base learners and online ensemble approaches such as Boosting and Random Subspace methods. Further experimentation will also be done with artificial datasets, with different kinds of concept drift induced at varying intervals.

References

1. Attar, V., Sinha, P., Wankhade, K.: A fast and light classifier for data streams. Evolving Systems 1(3), pp. 199-207 (2010)
2. Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavaldà, R.: New ensemble methods for evolving data streams. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 09, pp. 139-148, NY, USA (2009)
3. Bifet, A., Holmes, G., and Pfahringer, B.: Leveraging bagging for evolving data streams. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML/PKDD, pp. 135-150 (2010)
4. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive online analysis. J. Mach. Learn. Res., 11, pp. 1601-1604 (2010)
5. Breiman, L.: Bagging predictors. Mach. Learn., 24(2), pp. 23-140 (1996)
6. Brzezinski, D., Stefanowski, J.: Reacting to different types of concept drift: The accuracy updated ensemble algorithm. In: Neural Networks and Learning Systems, IEEE Transactions on, 25(1), pp. 81-94 (2014)
7. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 00, pp. 71 80, NY, USA (2000)
8. Gaber, M. M., Stahl, F., Gomes, J.B.: Pocket data mining: Big data on small devices, Studies in Big Data 2, Springer Verlag, (2014)
9. Hansen, L.K., Salamon, P.: Neural network ensembles. In: IEEE Trans. Pattern Anal. Mach. Intell., 12 (10), pp. 993-1001 (1990)
10. Kargupta, H., Hoon P., Pittie, S., Liu, L.: Mobimine: Monitoring the stock market from a PDA, ACM SIGKDD Explorations, 3, pp. 37-47, (2002)
11. Khoshgoftaar, T.M., Hulse, J.V., Napolitano, A.: Comparing boosting and bagging techniques with noisy and imbalanced data. In: IEEE Transactions on Systems, Man, and Cybernetics, Part A, 41(3), pp. 552-568 (2011)
12. Opitz, D., Maclin, R.: Popular ensemble methods: An empirical study. In: Journal of Artificial Intelligence Research, 11, pp. 169-198 (1999)
13. Oza, N.C., Russell, S.: Online bagging and boosting: In: Artificial Intelligence and Statistics, pp. 105–112 (2001)
14. Oza, N.C., Russell, S.: Experimental comparisons of online and batch versions of bagging and boosting. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, pp. 359-364 (2001)
15. van Rijn, J., Holmes, G., Pfahringer, B., Vanschoren, J.: Algorithm selection on data streams. Džeroski, S., Panov, P., Kocev, D., Todorovski, L. (eds.). In: Discovery Science, Lecture Notes in Computer Science, vol. 8777, pp. 325-336. Springer International Publishing (2014)
16. Žliobaite, I., Budka, M., Stahl, F.: Towards cost-sensitive adaptation: When is it worth updating your predictive model? In: Neurocomputing, 150, Part A(0), pp. 240–249 (2015)
17. Kolter, J.Z., Maloof, M.A.: Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts. In: Journal of Machine Learning Research 8, pp. 2755-2790 (2007)

Mining Periodic Changes in Complex Dynamic Data through Relational Pattern Discovery

Corrado Loglisci, Donato Malerba

Department of Computer Science, Universita' degli Studi di Bari "Aldo Moro", Italy
{corrado.loglisci,donato.malerba}@uniba.it

Abstract. The empowerment of the information technologies in many real-world applications has opened to the possibility of tracking complex and evolving phenomena and gather information able to describe such phenomena. For instance, in bio-medical applications, we can monitor a patient and collect data that range from his clinical picture to the laboratory studies on biological products. In this scenario, studying the possible alterations manifested over time becomes thus relevant and, in life sciences, even determinant. In this paper, we investigate the task of determining changes which are regularly repeated over time and we propose a method based on two notions of patterns, *emerging patterns* and *periodic changes*. The method works on a time-window model to the end of *i)* capturing statistically evident changes and *ii)* detecting their periodicity. We tested the method on real-world complex dynamic data collected from the phenotypes of virus of the influenza A/H1N1.

1 Introduction

The advances in the development of hardware devices and information technologies has augmented the complexity of the applications in fields such as life sciences, engineering and social sciences. The most prominent result lies in the possibility of following and dealing with complex phenomena, whose investigation passes through the analysis of the data that being generated and collected. Such data are high heterogeneous, characterized by different properties and composed of several entities. Another degree of complexity is represented by the fact that these applications often work in evolving scenarios, which leads to generate data dynamic in nature. The analysis of complex and time-changing data can provide us with tools able to deal with unexpected behaviors and understanding the underlying dynamics. As proof of that, we can mention an extensive list of techniques of data mining focused on complex dynamic data.

Mining changes in complex data is not a problem of immediate solution and we could not directly apply traditional methods to such data because the existing approaches would tend to oversimplify the different sources of information of the complex data. Indeed, changes in complex data can be originated from multiple and different factors, such as the structure, descriptive properties, entities constituting the main complex objects. This seems to be the same consideration which has stimulated recent works of change mining on structured evolving

data. For instance, the method proposed in [13] analyzes dynamic networks in order to detect changes occurred at the level the labels of the edges, while in [12], the same authors consider again dynamic networks but with the different task of identifying changes in the number of the occurrences of sub-networks. Changes can be searched also as variations of the global and local properties [7] of evolving graphs.

In many applications, changes can be unpredictable, they can reveal unexpected variations or can be even evolutions already observed in past which are newly repeated. The analysis of repetitive behaviors exhibited over time is not a novel problem and the literature on the periodicity detection contributes to it. The blueprint for most algorithms follows a frequent pattern-based framework according which repetitive behaviors can be identified as regularly (periodically) repeated sub-sequences in a lengthy sequence [9]. A common aspect to these works is that the periodic repetitions concerns stationary data (e.g., sub-sequences) that are replicated over time, while no attempt has been done for detecting periodic changes, that is, dynamic behaviors which regularly recur.

The identification of periodic changes cannot be trivially faced by resorting to any algorithm of discovery of periodic patterns due to several reasons. First, the periodic patterns describe repeated behaviors which are static, thus their periods would not refer to changes. Second, searching changes among the periodic patterns could lead to work on a restricted search space. Third, different periodic patterns could depict changes but their periods could be not aligned, therefore the problem would require further computation to determine the periodicity of the changes from the periodicity of the patterns. Fourth, the existing algorithms work on a sequence-based representation, which, in the case of complex data, tends to over-simplify the multiple and different aspects of the data with the risk of omitting periodic behaviors and changes originating from the inner entities.

In this paper, we investigate the task of capturing statistically evident changes emerged over time and tracking their repeatability. The proposed method adopts a model of analysis based on time-windows and uses a frequent pattern mining framework as mean for abstracting and summarizing the data. This enables us to search changes as differences between frequent patterns. Since frequency denotes regularity, patterns can provide empirical evidence about real changes. Frequent patterns are discovered from the complex data collected by time-windows and thus they reflect co-occurrences in terms of structure, properties and inner entities of the complex objects that are frequent in specific intervals of time. The changes which can emerge in this setting regard differences between the frequent patterns of two time-windows. In particular, we are interested in changes which are manifested as significant variations of the frequency of the patterns from a time-window to the next one. Not all the changes are considered, but only those which are replicated over time. We extend the concept of *Emerging Patterns* in order to depict changes between two time-windows and introduce the notion of *Periodic Changes* in order to characterize changes regularly repeated over time.

The paper is organized as follows. In Section 2 we report necessary notions, while the method is described in Section 3. An application to the phenotype data

in Virology is described in Section 4. Then, we overview the related literature (Section 5) and finally conclusions close the paper (Section 6).

2 Basics and Definitions

Most of the methods reported in the literature overcome the difficulty of accounting for the multiple and different aspects of the complex data objects with formalisms based on vectors or attribute-value sets, which model only global properties of the data. These solutions could be too limiting because they neglect the intrinsic structure of the complex data, entities that constitute the main complex object and the inner relationships existing among the entities. To overcome this drawback, we use the (multi-) relational setting [6], which has been argued to be the most suitable formalism for representing complex data.

In the relational setting, complex data and the constituting entities can play different roles in the analysis. We can distinguish them between target objects (TOs) and non-target objects ($NTOs$). The former are the main subjects of the analysis, while the latter are objects relevant for the current problem and associated with the former.

Let $\{t_1 \dots t_n\}$ be a a sequence of time-points. At each time-point t_i, a set of instances (TOs) is collected. A *time-window* τ is a sequence of consecutive time-points $\{t_i, \dots, t_j\}$ ($t_1 \leq t_i$, $t_j \leq t_n$) which we denote as $[t_i; t_{i+w}]$. The width w of a time-window $\tau = \{t_i, \dots, t_{i+w}\}$ is the number of time-points in τ, i.e. $w = j - i + 1$. Two time-windows τ and τ' are *consecutive* if $\tau = \{t_i, \dots, t_{i+w-1}\}$ and $\tau' = \{t_{i+w} \dots t_{i+2w-1}\}$. Two pairs of consecutive time-windows (τ,τ') and (τ'',τ''') are *δ-separated* if $(j + w) - (i + w) \leq \delta$ ($\delta > 0$, $\delta \geq w$,) with $\tau = \{t_i, \dots, t_{i+w-1}\}$, $\tau' = \{t_{i+w} \dots t_{i+2w-1}\}$, $\tau'' = \{t_j \dots t_{j+w-1}\}$, and $\tau''' = \{t_{j+w} \dots t_{j+2w-1}\}$). Two pairs of consecutive time-windows (τ,τ') and (τ'',τ''') are *chronologically ordered* if $(j + w) > (i + w)$. We assume that all the time-windows have the same width w and we use the notation τ_{h_k} to refer to a time-window and the notation (τ_{h_1}, τ_{h_2}) to indicate a pair of consecutive time-windows.

Both TOs and $NTOs$ can be represented in Datalog language as sets of ground atoms [3]. A ground atom is an n-ary logic predicate symbol applied to n constants. We consider three categories of logic predicates: 1)*key predicate*, which identifies the TOs, 2)*property predicates*, which define the value taken by a property of a TO or of a $NTOs$, and 3)*structural predicates*, which relate TOs with their $NTOs$ or relate the $NTOs$ each other.

In the following, we report an example of target object in the domain of virology. Virology is one of the fields in which rich sets of complex dynamic data can be collected.

virus(p1). epidemiological_condition(p1,enhanced_Trasmission_to_Human). epidemiological_condition(p1,severity). epidemiological_condition(p1,increased_Virulence)). clinical_condition(p1,adamantane_resistance). clinical_condition(p1,oseltamivir_resistance). clinical_condition(p1,polybasic_HA_Cleavage). dependent_by(adamantane_resistance,mutation). variation_of(mutation,protein_M2). is_a(mutation,v26F).

where

- *virus()* is the key predicate which identifies the TO named as *p1*;
- *epidemiological_condition(,), clinical_condition(,)* are structural predicates which relates the TO with the $NTOs$ identified as *enhanced_Trasmission_to_Human*, *severity*, *increased_Virulence*, *adamantane_resistance*, *oseltamivir_resistance*, *polybasic_HA_Cleavage*;
- *dependent_by(,)* is a structural predicate which relates the NTO *adamantane_resistance* with the NTO identified as *mutation*;
- *variation_of(,)* is a structural predicate which relates the NTO *mutation* to the NTO identified as *protein_M2*
- *is_a(,)* is a property predicate which relates the NTO *mutation* to the value v26F

The following definitions are crucial for this work:

Definition 1. Relational pattern

A conjunction of atoms $P = p_0(t_0^1), p_1(t_1^1, t_1^2), p_2(t_2^1, t_2^2), \ldots, p_m(t_m^1, t_m^2)$*, is a* relational pattern *if* p_0 *is the key predicate,* p_i*,* $i = 1, \ldots, m$ *is either a structural predicate or a property predicate.*

Terms t_i^j are either constants, which correspond to values of property predicates, or variables, which identify TOs or $NTOs$. Moreover, all variables are linked to the variable used in the key predicate (according to the linkedness property [14]).

A relational pattern P is characterized by a statistical parameter, namely the *support* (denoted as $sup_{\tau_{h_k}}(P)$), which denotes the relative frequency of P in the time-window τ_{h_k}. It is computed as the number of TOs of τ_{h_k} in which P occurs divided the total number of TOs of τ_{h_k}. When the support exceeds a minimum user-defined threshold $minSUP$, P is said to be *frequent.*

Definition 2. Emerging pattern-EP

Let (τ_{h_1}, τ_{h_2}) *be a pair of consecutive time-windows;* P *be a frequent relational pattern in* τ_{h_1} *and in* τ_{h_2}*;* $sup_{\tau_{h_1}}(P)$ *and* $sup_{\tau_{h_2}}(P)$ *be the support of the pattern* P *in* τ_{h_1} *and in* τ_{h_2}*.* P *is an emerging pattern in* (τ_{h_1}, τ_{h_2}) *iff* $\frac{sup_{\tau_{h_1}}(P)}{sup_{\tau_{h_2}}(P)} \geq minGR \quad \vee \quad \frac{sup_{\tau_2}(P)}{sup_{\tau_1}(P)} \geq minGR$

where, $minGR$ (> 1) is a user-defined minimum threshold. The ratio $sup_{\tau_{h_1}}(P)/sup_{\tau_{h_2}}(P)$ is denoted with $GR_{\tau_{h_1},\tau_{h_2}}(P)$ and it is called *growth-rate* of P from τ_{h_1} to τ_{h_2}. When $GR_{\tau_{h_1},\tau_{h_2}}(P)$ exceeds $minGR$, we have that the support of P decreases from τ_{h_1} to τ_{h_2} by a factor equal to the ratio $sup_{\tau_{h_1}}(P)/sup_{\tau_{h_2}}(P)$, while when $GR_{\tau_{h_2},\tau_{h_1}}(P)$ exceeds $minGR$, the support of P increases by a factor equal to $sup_{\tau_{h_2}}(P)/sup_{\tau_{h_1}}(P)$.

The concept of emerging patterns is not novel in the literature [5]. In its classical formulation, it refers to the values of support of the same pattern which

has been discovered in two different classes of data, while, in this work, we extend it to represent the differences between the data collected in two intervals of time, and therefore, we refer to the values of support of the same pattern which has been discovered in two time-windows.
In the following, we report an example of emerging pattern.

P: phenotype(H),clinical_condition(H,C),dependent_by(C,M),variation_of(M,N).

with τ_{h_1}=[1991;1995], τ_{h_2}=[1996;2000], $sup_{[1991;1995]}(P) = 0.8$ and $sup_{[1996;2000]}(P) = 0.5$. Here, the support of the pattern P decreases, whereby of the growth-rate $GR_{[1991;1995],[1996;2000]}(P)$ is 1.6 (0.8/0.5). By supposing that $minGR$=1.5, the pattern P is considered emerging in ([1991;1995],[1996;2000]).

Definition 3. Periodic change-PC

Let $T : \langle(\tau_{i_1}, \tau_{i_2}), \ldots, (\tau_{m_1}, \tau_{m_2})\rangle$ be a set of chronologically ordered pairs of time-windows; P be an emerging pattern in all the pairs (τ_{h_1}, τ_{h_2}) with $h = i, \ldots, m$; $\langle GR_{\tau_{i_1},\tau_{i_2}}, \ldots, GR_{\tau_{m_1},\tau_{m_2}}\rangle$ be the values of growth-rate of P in the pairs $\langle(\tau_{i_1}, \tau_{i_2}), \ldots, (\tau_{m_1}, \tau_{m_2})\rangle$ respectively; $\Theta_P : \Re \rightarrow \Psi$ be a function which maps $GR_{\tau_{h_1},\tau_{h_2}}(P)$ into a discrete value $\psi_{\tau_{h_1},\tau_{h_2}} \in \Psi$ with $h = i, \ldots, m$. P is a periodic change iff:

1. $|T| \geq minREP$
2. (τ_{h_1}, τ_{h_2}) *and* (τ_{k_1}, τ_{k_2}) *are δ-separated for all* $h = i, \ldots, m-1$, *k=h+1 and there is no pair* (τ_{l_1}, τ_{l_2}), $h < l$, *s.t.* (τ_{h_1}, τ_{h_2}) *and* (τ_{l_1}, τ_{l_2}) *are δ-separated*
3. $\psi = \psi_{\tau_{i_1},\tau_{i_2}} = \ldots = \psi_{\tau_{m_1},\tau_{m_2}}$

where $minREP$ is a user-defined threshold. A PC is a frequent pattern whose support increases (decreases) at least $minREP$ times with an order of magnitude greater than $minGR$. Each change (increase/decrease) occurs within δ time-points and it is characterized by the value ψ. Intuitively, a PC represents a variation, manifested with a particular periodicity, of the frequency of the same pattern.

Note that Definition 3 uses the value δ as maximum threshold of periodicity. This leads to a two-fold result. On one hand, the changes could not be perfectly periodic, that is, the EP associated with a PC could be repeated even with a distance less than δ time-points. On the other hand, this allows us to discover a larger set of periodic changes that includes also those asynchronous [10], that is, changes with some disturbances between the repetitions.

An example of PC is reported here. Consider the following EPs

P: phenotype(H),clinical_condition(H,C),dependent_by(C,M), variation_of(M,N)
emerging in ([1991;1992],[1993,1994])
P: phenotype(H),clinical_condition(H,C),dependent_by(C,M), variation_of(M,N)
emerging in ([1996;1997],[1998;1999])
P: phenotype(H),clinical_condition(H,C),dependent_by(C,M), variation_of(M,N)
emerging in ([1999;2000],[2001;2002])

P: phenotype(H),clinical_condition(H,C),dependent_by(C,M), variation_of(M,N)
emerging in ([2004;2005],[2006,2007])

Here, $\psi_{[1991;1992],[1993;1994]} = \psi_{[1996;1997],[1998;1999]} = \psi_{[2004;2005],[2006:2007]}$, $\psi_{[1991;1992],[1993;1994]} \neq \psi_{[1999;2000],[2001;2002]}$. By supposing $minREP$=2 and $\delta = 6$, P is a periodic change . Indeed, $T : \langle([1991; 1992], [1993; 1994]), ([1996; 1997], [1998, 1999])\rangle$ meets the conditions (1) and (2) because $|T| = 2$ and (1998-1993)<6; the discrete values of the growth-rate in ([1991;1992],[1993;1994]) and ([1996;1997],[1998;1999]) meet the condition (3). The pair of time-windows ([1999; 2000], [2001; 2002]) is not considered because $\psi_{[1999;2000],[2001;2002]}$ does meet the condition (3), while the pair of time-windows ([2004; 2005],[2006; 2007]) does not meet the condition (3) because (2006-1998)>6.

3 The Algorithm

We propose an algorithm which discovers PCs incrementally as time goes by. It works on the succession $\langle(\tau_{1_1}, \tau_{1_2}), \ldots, (\tau_{h_1}, \tau_{h_2}), \ldots\rangle$ of pairs of time-windows obtained from $\{t_1, \ldots, t_n\}$. Each time-window τ_{u_v} (except that for the first and last one) is present in two pairs, that is, the pair (τ_{h_1}, τ_{h_2}) where $\tau_{u_v} = \tau_{h_2}$, and the pair $(\tau_{(h+1)_1}, \tau_{(h+1)_2})$ with $\tau_{u_v} = \tau_{(h+1)_1}$. This is done with the intent to capture the changes of support of the patterns from τ_{h_1} to τ_{u_v} and from τ_{u_v} to $\tau_{(h+1)_2}$. For each pair of time-windows (τ_{h_1}, τ_{h_2}), the algorithm performs three steps: *1)* Discovery of frequent patterns on the time-windows τ_{h_1} and τ_{h_2} separately; *2)* Extraction of EPs by matching the frequent patterns discovered from τ_{h_1} against the frequent patterns discovered from τ_{h_2}. These EPs are stored in a pattern base, which is incrementally updated as the time-windows are processed; *3)* Detection of PCs by testing the conditions of Definition 3 on the EPs stored in the base. Note that, when the algorithm processes the pair $(\tau_{(h+1)_1}, \tau_{(h+1)_2})$, it uses the frequent patterns of the time-window τ_{h_2}, which had been discovered when the algorithm had processed the pair (τ_{h_1}, τ_{h_2}). This avoids of performing the step *1)* twice on the same time-window. Details on these three steps are reported in the following.

3.1 Relational frequent pattern discovery

Frequent patterns are discovered from each time-window by following the method proposed in [1], which enables the discovery of relational patterns whose support exceeds $minSUP$. It explores level-by-level the lattice of the patterns, from the most general to the more specific ones, starting from the most general pattern (which contains only the key predicate). The lattice is organized according to a generality ordering based on the notion of θ-subsumption [14]. Formally, given two relational patterns $P1$ and $P2$, $P1$ ($P2$) is more general (specific) than $P2$ ($P1$) under θ-subsumption, denoted as $P1 \geqslant_\theta P2$, if and only if $P2$ θ-subsumes $P1$, where $P2$ θ-subsumes $P1$ if and only if a substitution θ exists such that $P2\ \theta \subseteq P1$. The method implements a two-stepped procedure: *i)* generation of candidate patterns with k atoms (k-th level) by considering the frequent patterns

with $k-1$ atoms (*(k-1)*-th level); *ii)* evaluation of the support of the patterns with k atoms.

The monotonicity property of the support value (i.e., a super-set of an non-frequent pattern cannot be frequent) is exploited to avoid the generation of non-frequent relational patterns. In fact, in accordance with the Definition 2, non-frequent patterns are not used for detecting changes and thus we can prune portions of the space containing non-frequent patterns. Thus, given two relational patterns $P1$ and $P2$ with $P1 \geqslant_\theta P2$, if $P1$ is non-frequent in a time-window, then the support of $P2$ is less than the threshold $minSUP$ and it is non-frequent too in the same time-window. Therefore, we do not refine the patterns which are non-frequent.

3.2 Emerging pattern extraction

Once the frequent patterns have been discovered from the time-windows τ_{h_1} and τ_{h_2}, they are evaluated in order to check if the growth-rate exceeds the threshold $minGR$. Unfortunately, the monotonicity property does not hold for the growth-rate. In fact, given two frequent patterns $P1$ and $P2$ with $P1 \geqslant_\theta P2$, if $P1$ is not emerging, namely $GR_{\tau_{h_1},\tau_{h_2}}(P1) < minGR$ ($GR_{\tau_{h_2},\tau_{h_1}}(P1) < minGR$), then the pattern $P2$ may or may not be an EP, namely its growth-rate could exceed the threshold $minGR$. However, we can equally optimize this step by avoiding the evaluation of the refinements of a pattern P discovered from the time-window τ_{h_1} (τ_{h_2}) in the case P is non-frequent in the time-window τ_{h_2} (τ_{h_1}). Note that this operation could exclude EPs with very high values of growth-rate (i.e., the strongest changes), but here we are interested in the changes exhibited by co-occurrences which are statistically evident in both intervals of time.

The EPs extracted on the pairs of time-windows are stored in the pattern base, which hence contains the frequent patterns that satisfy the constraint set by $minGR$ on at least one pair of time-windows. Each EP is associated with two lists, named as $TWlist$ and $GRlist$. $TWlist$ is used to store the pairs of time-windows in which the growth-rate of the pattern exceeds $minGR$, while $GRlist$ is used to store the corresponding values of growth-rate. To distinguish the changes due to the decrease of the support from those due to the increase, we store the values of growth-rate as negative (that is, with the minus sign) when it decreases.

The base is maintained with two operations, namely insertion of the EPs and update of the lists $TWlist$ and $GRlist$ associated with the EPs. A pattern is inserted if it has not been recognized as emerging in the previous pairs of time-windows, while, if it has been previously inserted, we update the two lists.

3.3 Periodic change detection

The step *3)* works on the pattern base and filters out the EPs that do not meet the conditions of Definition 3. The function Θ_P implements an equal-width discretization technique, which is applied to two sets of values obtained from the lists $GRlist$ of all the stored EPs, the first set consists of all the positive values

of growth-rate, the second one consists of all the negative values. Note that we have not infinite values of growth-rate because all the patterns considered are frequent, i.e., there are no values of support equal to zero. The ranges returned by the discretization technique correspond to the discrete values $\psi_{\tau_h,\tau_{h+1}}$. Thus, we have two sets of ranges Ψ^+ and Ψ^-: Ψ^+ refers to the discrete values obtained from the positive values of growth-rate, while Ψ^- refers to the discrete values obtained from the negative values. We replace the numeric values contained in the lists *GRlist* with the corresponding ranges in Ψ^+ and Ψ^-. This allows us to obtain two separate sets of discrete values and capture the increases/decreases of the support of the patterns by representing them with a finite number of cases.

This new representation of the growth-rate could suggest to prune the EPs that are more general and conserve the EPs that are more specific when they have the same discrete values. But, this cannot be done because it is not guaranteed that there is equality between the discrete values over all the time-windows.

In this step, the algorithm performs two preliminary operations: i) removal of the EPs when their lists *TWlist* and *GRlist* have length less than the threshold *minREP*; ii) sorting the remaining lists *TWlist* by chronological order. The lists *GRlist* will be re-arranged accordingly.

The algorithm discovers PCs by working on the EP separately and it can identify more than one PC from a single EP. For each EP, it scans the *TWlist* once and incrementally builds the set T of each candidate PC. A candidate PC is characterized by one discrete value. During the scan, it evaluates the current pair of time-windows (τ_h, τ_{h+1}) of *TWlist* and the relative discrete value $\psi_{\tau_h,\tau_{h+1}}$ against with the latest pair of time-windows (τ_k, τ_{k+1}) inserted in the set T of the candidate PC that has the same discrete value: if the pairs of time-windows are δ-separated, then the pair (τ_h, τ_{h+1}) in inserted in the set T of the candidate PC, otherwise it can be considered to start the construction of the set T of a new candidate PC having the same discrete value. Finally, the algorithm filters our the PCs with $|T|$ less than the threshold *minREP*.

In order to clarify how the step *3)* works, we report an explanatory example. Consider $\Psi^+ = \{\psi', \psi''\}$, *minREP*=3, δ=13 and the lists *TWlist* and *GRlist* built as follows:

$$TWlist : \langle([1970;1972],[1973;1975]),([1976;1978],[1979;1981]),([1982;1984],[1985,1987]),$$
$$([1988;1990],[1991;1993]),([1994;1996],[1997;1999]),([2010;2012],[2013;2015])\rangle$$
$$GRlist : \langle \quad \psi', \quad \psi', \quad \psi'', \quad \psi', \quad \psi'', \quad \psi' \rangle$$

By scanning the list *TWlist*, we can initialize the set T of a candidate PC' by using the pairs ([1970;1972],[1973;1975]) and ([1976;1978],[1979;1981]) since they are δ-separated (1979-1973< δ) and they have the same discrete value ψ'. The pair ([1982;1984],[1985;1987]) instead refers to a different discrete value (ψ'') and therefore it cannot be inserted into T of PC'. We use it to initialize the set T of a new candidate PC", which thus will include the time-windows referred to ψ''. Subsequently, the pair ([1988;1990],[1991;1993]) is inserted into T of PC' since its distance from the latest pair is less than δ (1991-1979< δ). Then, T of PC" is updated with ([1994;1996],[1997;1999]) since 1997-1985 is less than δ, while the pair ([2010;2012],[2013;2015]) cannot be inserted into T because

the distance between 2013 and 1997 is greater than δ. Thus, we use the pair ([2010;2012],[2013;2015]) to initialize the set T of a new candidate PC"'. The set T of PC' cannot be further updated, but, since its size exceeds $minREP$, we consider the candidate PC' as valid periodic change. Finally, the candidate PC" cannot be considered as valid since its size is less than $minREP$. The candidate PC"' is not even considered since $|T_{\psi'}| < minREP$.

4 Experiments

We prove the viability of the proposed method on the data concerning the phenotypes of the influenza A/H1N1 virus in Virology. Studying the alterations of the phenotypes on a temporal basis is relevant and even determinant whether considering the biological re-assortment between the involved organisms and the cyclic nature of the pandemic outbreaks. The influenza A virus can infect several species. The sources of complexity of such data are several. In particular, the virus contains eight segments gene of negative single-stranded RNA, namely $PB2$, $PB1$, PA, HA, NP, NA, M, and NS encoding for 11 proteins. Moreover, the subtype of influenza A virus is determined by the antigenicity (the capacity to induce an immune response) of the two surface glycoproteins, haemagglutinin (HA) and neuraminidase (NA). Distinct subtypes determining different clinical and epidemiological conditions [8].

The datasets we use comprise phenotype data describing isolate strains of viruses of three different species, i.e., human, avian and swine. These isolate strains have been registered from 1958 to September 2009, while the datasets have been generated as a view on Influence Research Database hosted at the NIAID BioHealthBase BRC[1] and contain 3221 isolate strains for human, 1119 isolate strains for swine, and 757 isolate strains for avian.

Experiments are performed to study the effect of the thresholds w, δ, $minGR$ and $minREP$ on the discovered periodic changes and emerging patterns. The parameter $minSUP$ is fixed to 0.1. In this case study, the time-points correspond to years, while the number of the ranges produced by the discretization function is fixed to 5. Statistics on the results are collected in Table 1.

In Table 1(a), we have the number of PCs and EPs when tuning w (δ=20, $minGR = 2$, $minREP = 3$). By increasing the width of the time-windows, the overall number of the time-windows decreases, which results in a shorter succession of pairs of time-windows where finding EPs and PCs. This explains the decrease of the number of EPs and PCs for swine and human. We have a different behavior for the phenotypes of avian, where the number of EPs and PCs increases. Indeed, the use of wider time-windows (w=10 and 15 against w=5 and 7) leads to collect greater sets phenotypes having likely higher changeability. In this case, it seems that phenotype change concerns longer periods.

In Table 1(b), we have the results when tuning δ (w=5, $minGR = 2$, $minREP = 3$). As expected, higher values of δ allow us to detect a more numerous set of PCs, which comprises both the replications of EPs which are closer

[1] `http://www.fludb.org/brc/home.do?decorator=influenza`

and the replications of EPs that are distant. Whilst, when δ is 10, we capture only the PCs that cover at most ten years. The threshold δ does not affect the number of EPs since it operates after the extraction of the EPs. In Table 1(c), we have the results when tuning $minGR$ (w=5, δ=20, $minREP = 3$). We observe that $minGR$ has great effect on the number of PCs and on the number of EPs. Indeed, at high values of $minGR$, the algorithm is required to detect the strongest changes of support of the patterns, which leads to extract only the EPs with the higher values of growth-rate. This explains the decrease of the number of PCs. The threshold $minREP$ has no effect on the EPs since it acts on the PCs only (Table 1(d), w=5, δ=20, $minGR = 2$). As expected, higher values of $minREP$ lead to exclude the EPs that have a low number of replications. This means that the algorithm works on a smaller set of EPs, with the result to have a lower number of PCs. In particular, when $minREP$ is 6, we have no PC that includes EPs repeated six times and distant at most 20 years.

In the following, we report an example of periodic change discovered by the proposed algorithm from the human dataset with w=10, δ=20, $minGR = 2$, $minREP$=3:

phenotype(P), epidemiological_condition(P,E), is_a(E,enhanced_Transmission_to_Human), dependent_by(E,M1), is_a(M1,mutation_A199S), mutation_of(M1,T), is_a(T,protein_PB2), dependent_by(E,M2), is_a(M2,mutation_A661T), mutation_of(M2,T), dependent_by(E,M3), is_a(M3,mutation_K702R), mutation_of(M3,T).

Here, $T : \langle([1958; 1967], [1968; 1977]), ([1968; 1977], [1978; 1987]), ([1988; 1997], [1998; 2009])\rangle$, ψ=[2;3,5]. The periodic change concerns the epidemiological condition 'enhanced_Transmission_to_Human' with the mutations 'A199S' on the protein 'PB2' and the mutations 'A661T' and 'K702R'. Indeed, the frequency of this pattern increases three times by a factor included in the range [2;3,5]. This happens between the time-windows [1958;1967] and [1968;1977], [1968;1977] and [1978;1987], [1988;1997] and [1998;2009].

	w(years)			
	5	7	10	15
swine	11–126	11–126	8–63	2–18
human	20–176	20–176	10–69	2–44
avian	1–4	1–4	2–10	2–10

(a)

	δ(years)			
	10	15	20	25
swine	0–126	6–126	11–126	15–126
human	14–176	18–176	20–176	22–176
avian	0–4	0–4	1–4	1–4

(b)

	$minGR$			
	2	4	6	8
swine	11–126	7–68	0–2	0–0
human	20–176	6–61	0–4	0–0
avian	1–4	0–2	0–2	0–0

(c)

	minREP			
	3	4	5	6
swine	11–126	6–126	2–126	0–126
human	20–176	7–176	3–176	0–176
avian	0–6	0–2	0–2	0–2

(d)

Table 1. Total number of the periodic changes and emerging patterns discovered on the three species when tuning the width of the time-window w (a), the maximum admissible distance between consecutive pairs of time-windows δ (b), minimum threshold of growth-rate $minGR$ (c) and minimum threshold of repetitions $minREP$ (d). In each cell, we have reported the statistics as number of PCs–number of the EPs.

5 Related Works

The studies on the periodicity have been concentrated on the identification of the occurrences of static behaviors over data sequence. Most of proposed methods follow the frequent pattern mining framework and can be categorized mainly on the basis of the notion of periodicity. The periodicity is asynchronous when the occurrences are not always regular [10], while, we have a partial periodicity [4] when the repetitions do not exhibit always the same elements. Other works have been focused on the periodicity in complex data. For instance, in [11], the authors report an algorithm to find a minimal set of periodically recurring subgraphs that capture all periodic behavior in a dynamic network. They rely on the concepts of closed subgraphs and Occam's razor.

However, at our knowledge, we cannot enumerate attempts on the study of the periodicity of changing behaviors, despite to the recent interest in change mining problems. In this sense, two main research lines can be distinguished. In the first line, we find methods to identify changes in the global properties of the complex data. For instance, in [2] the authors describe a hierarchical clustering technique to identify eras of evolution of a dynamic network. An era is associated to a cluster and it is produced as a sequence of structurally similar temporal snapshots of the network, thus a new cluster represents a structural change with respect to previously generated clusters and denotes the beginning of a new era. In the second line, we find methods to characterize evolutions of local properties. For instance, in [13], we proposed an algorithm for capturing variations exhibited from the patterns discovered from data network over time. Two different notions of changes were formalized, namely change patterns and change chains, the first one denotes the evolution of the network from a time-period to the next one, the second one denotes the whole evolution over the temporal axis.

6 Conclusions

In this paper we investigated two different aspects of the analysis of complex dynamic data, namely the discovery of changes manifested over time and identification of those changes that repeat over time with a certain periodicity. The proposed method relies on a frequent pattern mining framework which enable us to *i)* capture the changes with variations of the frequency of frequent patterns and *ii)* determine the periodicity of the changes with regularly recurring variations of the frequency. At this end, we extended the classical notion of emerging patterns and formalized the new notion of periodic change. A periodic change is an emerging pattern that meets the constraint of periodicity set as input parameter, that is, the maximum period with which the emerging pattern recurs is known. However, often changes can repeat with unexpected periods, hence using a known value of periodicity could miss some patterns. This is the main future direction we intend to upgrade the method. The experiments highlights the viability of the proposed method to real-world problems in Virology. We are confident that the method could be used for similar problems in other scenarios.

Acknowledgements. The authors would like to acknowledge the support of the European Commission through the project MAESTRA - Learning from Massive, Incompletely annotated, and Structured Data (Grant number ICT-2013-612944).

References

1. A. Appice, M. Berardi, M. Ceci, and D. Malerba. Mining and filtering multi-level spatial association rules with ARES. In *Foundations of Intelligent Systems, 15th International Symposium, Proceedings*, pages 342–353, 2005.
2. M. Berlingerio, M. Coscia, F. Giannotti, A. Monreale, and D. Pedreschi. Evolving networks: Eras and turning points. *Intelligent Data Analysis*, 17(1):27–48, 2013.
3. S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer, 1990.
4. S. Chen, T. C. Huang, and Z. Lin. New and efficient knowledge discovery of partial periodic patterns with multiple minimum supports. *Journal of Systems and Software*, 84(10):1638–1651, 2011.
5. G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 43–52, 1999.
6. S. Džeroski and N. Lavrač. *Relational Data Mining*. Springer-Verlag, 2001.
7. J. Ferlez, C. Faloutsos, J. Leskovec, D. Mladenic, and M. Grobelnik. Monitoring network evolution using MDL. In G. Alonso, J. A. Blakeley, and A. L. P. Chen, editors, *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, México*, pages 1328–1330. IEEE, 2008.
8. Y. Furuse, A. Suzuki, T. Kamigaki, and H. Oshitani. Evolution of the m gene of the influenza a virus in different host species: large-scale sequence analysis. *Virology Journal*, 6(67), 2009.
9. J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. In *Proceedings of the 15th International Conference on Data Engineering, Sydney, Austrialia, March 23-26, 1999*, pages 106–115, 1999.
10. K. Huang and C. Chang. SMCA: A general model for mining asynchronous periodic patterns in temporal databases. *IEEE Trans. Knowl. Data Eng.*, 17(6):774–785, 2005.
11. M. Lahiri and T. Y. Berger-Wolf. Periodic subgraph mining in dynamic networks. *Knowl. Inf. Syst.*, 24(3):467–497, 2010.
12. C. Loglisci, M. Ceci, and D. Malerba. Discovering evolution chains in dynamic networks. In *New Frontiers in Mining Complex Patterns - First International Workshop, NFMCP 2012, Held in Conjunction with ECML/PKDD 2012, Bristol, UK, September 24, 2012, Revised Selected Papers*, pages 185–199, 2012.
13. C. Loglisci, M. Ceci, and D. Malerba. Relational mining for discovering changes in evolving networks. *Neurocomputing*, 150, Part A(0):265 – 288, 2015.
14. G. D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.

Applicability of Roughly Balanced Bagging for Complex Imbalanced Data

Mateusz Lango and Jerzy Stefanowski

Institute of Computing Science, Poznań University of Technology,
60-965 Poznań, Poland

Abstract. Roughly Balanced Bagging is based on under-sampling and classifies imbalanced data much better than other ensembles. In this paper, we experimentally study its properties that may influence its good performance. Results of experiments show that it can be constructed with a small number of component classifiers, which are quite accurate, however, of low diversity. Moreover, its good performance comes from its ability to recognize unsafe type of minority examples better than other ensembles. We also present how to improve its performance by integrating bootstrap sampling with random selection of attributes.

Keywords: class imbalance, ensembles, Roughly Balanced Bagging, types of minority examples

1 Introduction

Learning classifiers from imbalanced data still reveals research challenges. However, difficulties are not caused by the unbalanced class cardinalities only. Deterioration of classification performance arises when other data difficulty factors occur together with the class imbalance ratio, such as decomposition of the minority class into rare sub-concepts, too extensive overlapping of decision classes or presence of minority examples inside the majority class regions [9, 13].

Several methods have been introduced to deal with imbalanced data; for their review see, e.g., [5]. New specialized *ensembles* are able to handle complex imbalanced distributions better than simpler approaches; see their review in [4, 12]. They are usually modifications of bagging or boosting and either employ pre-processing methods before learning component classifiers or embed the cost-sensitive framework in the learning process. However, the comparative studies of new ensembles are still too limited. Studies [1, 4, 10] have showed that extensions of bagging work better than generalizations of boosting and more complex solutions. The recent study [2] demonstrated that Roughly Balanced Bagging [6] has achieved the best results and is significantly better than other over-sampling extensions of bagging.

The key idea behind Roughly Balanced Bagging is a specific random under-sampling before generating component classifiers, which reduces the presence of the majority class examples inside each bootstrap sample. Although this ensemble has been successfully used in several papers, there are not enough attempts

to check which of its characteristics are the most crucial for improving classification of complex imbalanced data. In our opinion, its properties should be examined more precisely.

The aim of this paper is to experimentally study the following issues: (1) the most influential aspects of constructing Roughly Balanced Bagging and its main properties (with respect to bootstrap construction, deciding on the number of component classifiers, their diversity, methods for aggregating predictions); (2) abilities of this ensemble to deal with different types of difficult distributions of the minority class; (3) directions for its further extension and improvements.

2 Related Works

For the reviews of ensembles dedicated for class imbalance consult [4, 5, 12]. Galar et al. in [4] distinguish mainly *cost-sensitive* approaches vs. integrations with *data pre-processing*. Below we briefly present under-bagging proposals only, which are most relevant to our study.

Breiman's algorithm for learning bagging samples a number of subsets from the training set, builds multiple base classifiers and aggregates their predictions to make a final decision. *Bootstraps* are generated by uniform random sampling with replacement of instances from the original training set (usually keeping the size of the original set). However, as this sampling is performed on all data elements, regardless their class labels (majority or minority), the imbalanced class distribution will be hold in each bootstrap and the ensemble will fail to sufficiently classify minority class. Most of current proposals overcome this drawback by applying pre-processing techniques to each bootstrap sample, which change the balance between classes – usually leading to the same, or similar, cardinality of the minority and majority classes. For instance, the over-sampling methods typically replicate the minority class data (either by random sampling or generating synthetic examples) to balance bootstraps.

In *under-bagging* the number of the majority class examples in each bootstrap is randomly reduced to the cardinality of the minority class (N_{min}). In the simplest proposals, as *Exactly Balanced Bagging*, the entire minority class is just copied to the bootstrap sample and then combined with the randomly chosen subset of the majority class to exactly balance cardinality between classes.

While such under-bagging strategies seem to be intuitive and work efficiently in some studies, Hido et al. [6] have claimed that they do not truly reflect the philosophy of bagging and could be still improved. In the original bagging the class distribution of each sampled subset varies according to the binomial distribution while in the above under-bagging each subset has the same class distribution as the desired balanced distribution. In *Roughly Balanced Bagging* (RBBag) the numbers of instances for both classes are determined in a different way by equalizing the sampling probability of each class. The number of minority examples (S_{min}) in each bootstrap is set to the size of the minority class N_{min} in the original data. In contrast, the number of majority examples is decided probabilistically according to the negative binomial distribution, whose parameters

are the number of minority examples (N_{min}) and the probability of success equal to 0.5. In this approach only the size of the majority examples (S_{maj}) varies, and the number of examples in the minority class is kept constant since it is small. Finally, component classifiers are induced by the same learning algorithm from each i bootstrap sample ($S^i_{min} \cup S^i_{maj}$) and their predictions form the final decision with the equal weight majority voting.

Hido et al. compared Roughly Balanced Bagging with several algorithms showing that it was better on G-mean and AUC measures [6]. The study [10] demonstrated that under-bagging, including RBBag, significantly outperformed best extensions of boosting and the difference was more significant when data were more noisy. The results of [2] showed that Roughly Balanced Bagging was significantly better than best oversampling extensions of bagging and usually better than Exactly Balanced Bagging. These experiments also supported using sampling with replacement in RBBag – so, we will also use it in further experiments. However, there are not so many attempts to either to experimentally examine properties of this ensemble or to more theoretically explain why and when it should outperform other methods. Only the work [16] provides a probabilistic theory of imbalance and its reference to under-sampling classifiers.

3 Studying the Role of Components in Roughly Balanced Bagging

The first part of experiments aims at studying the following basic properties of constructing Roughly Balanced Bagging, which have not been studied in the literature yet: (1) Using different learning algorithms to built component classifiers; (2) The influence of the number of component classifiers on the final performance; (3) The role of diversity of component classifiers.

We extend the previous implementation of RBBag done by L. Idkowiak for the WEKA framework [2]. We choose 24 UCI datasets which have been used in the most related experimental studies [3, 10, 13, 14]. They represent different imbalance ratios and other data difficulty factors - so they constitute various difficulty levels for ensembles [3, 14]. Moreover, we consider binary class versions of these data as it is done in the related studies [4, 6, 10]. For their detailed characteristics the reader is referred to [3].

The performance of ensembles is measured using: *sensitivity* of the minority class (the minority class accuracy), its *specificity* (the majority classes accuracy), their aggregation to the *geometric mean* (G-mean) and *F-measure*. For their definitions see, e.g., [5]. We have chosen these point measures, instead of AUC as the most of considered learning algorithms produce deterministic outputs. These measures are estimated with the stratified 10-fold cross-validation repeated several times to reduce the variance.

3.1 Choosing Algorithms to Learn Component Classifiers

The related works show that Roughly Balanced Bagging as well as other under-sampling extensions of bagging are usually constructed with decision trees. In

this study we check whether classification performance of this ensemble may depend on using other learning algorithms. Besides J48 unpruned tree we considered Naive Bayes tree, rule algorithms – Ripper and PART, Naive Bayes classifiers and SVM – all available in WEKA. The RBBag ensemble was constructed with different numbers (30, 50 and 70) of component classifiers.

Here, we summarize the Friedman test only. For all considered evaluation measures we were unable to reject the null hypothesis on equal performance of all versions of RBBag. For instance, average ranks in the Friedman test for G-mean (the smaller, the better) were the following: SVM 4.1; Ripper 4.12; NBTree 4.4; J48tree/PART 4.5; NB 4.8. Quite similar rankings were obtained for other measures. All these results did not show significant differences of using any of this algorithm inside RBBag.

Furthermore, for each single algorithm RBBag was significantly better than its standard bagging equivalent (according to the paired Wilcoxon test).

3.2 The Influence of the Number of Component Classifiers

Related works showed that Roughly Balanced Bagging was used with rather a high number of component classifiers. Hido et al. [6] tested it with 100 C4.5 trees. In the study [10] authors applied a dozen of components. Then, the study [2] showed that it also performed well with 30, 50 or 70 classifiers. Thus, we have decided to examine more systemically other (also smaller) sizes of this ensemble and its influence on the final performance. We stayed with learning components with J48 unpruned trees, and for each dataset we constructed a series of Roughly Balanced Bagging ensembles increasing its size one by one - so the number of component classifiers changed from 2 trees up to 100 ones.

For all considered datasets increasing the number of component classifiers improves the evaluation measures up to the certain size of the ensemble. Then, values of measures stay at a stable level or slightly vary around the certain level. Due to page limits, in Fig. 1 we present the most representative changes of G-mean values. Figures for few other datasets present similar trends.

What is even more surprised the RBBag ensemble achieves this good performance for a relatively small number of component classifiers. For most datasets, the stable highest value of G-mean is observed approximately between 10 and 15 trees. In case of the sensitivity or F-measure we noticed similar tendencies.

Moreover, we decided to examine confidence of the final decision of RBBag. We refer to a *margin* of the ensemble prediction. For standard ensembles, it is usually defined as a difference between the number of votes of component classifiers for the most often predicted class label and the number of votes for the second predicted label. Here, we modified it as: $marg = (n_{cor} - n_{incor})/n_{cptclas}$, where n_{cor} is the number of votes for the correct class, n_{incor} is the number of votes for the incorrect class and $n_{cptclas}$ is the number of component classifiers in the ensemble. The higher absolute value of $marg$ is interpreted as high confidence while values closer to 0 indicate uncertainty in making a final decision for a classified instance. In Fig 2 we present a representative trend of changes of the relative margin with the size of RBBag for `ecoli` and `cmc` data. For many other

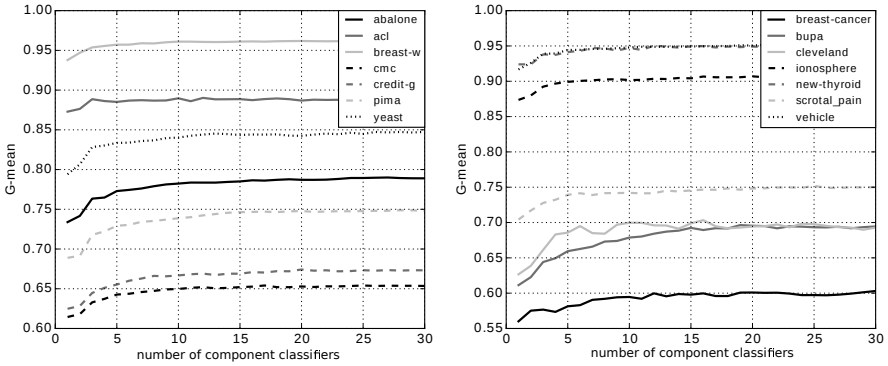


Fig. 1. G-mean vs. a number of component classifiers in RBBag for selected datasets.

datasets the trend line of the margin also stabilizes after a certain size (Note the resolution of the margin scale is more detailed than G-mean, so margin values achieve a satisfactory level also quite fast). We can conclude that the good performance of Roughly Balanced Bagging comes from rather a small number of component classifiers.

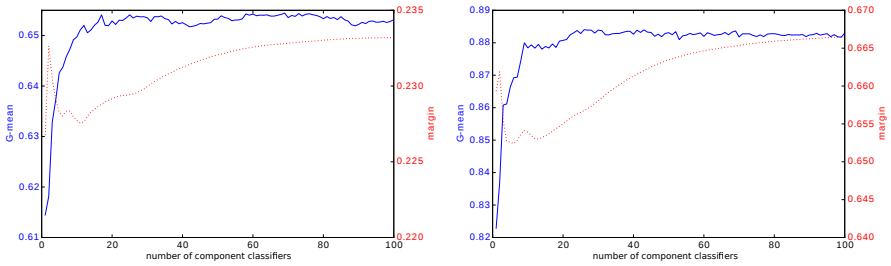


Fig. 2. G-mean and margin vs. a number of component classifiers in RBBag for `cmc` (left) and `ecoli` (right) datasets.

3.3 Diversity of Component Classifiers

The final accuracy of ensembles may be also related to their diversity - which is usually understood as the degree to which component classifiers make different decisions on one problem (in particular, if they do not make the same wrong decisions). Although, such an intuition behind constructing diverse component classifiers is present in many solutions, research concerns the total accuracy perspective [11]. It is still not clear how diversity affects classification performance

especially on minority classes. The only work on ensembles dedicated for imbalance data [17] does not provide a clear conclusion. Its authors empirically studied diversity of specialized over-sampling ensembles and noticed that larger diversity improved recognition of the minority class, but at the cost of deteriorating the majority classes. However, nobody has analysed diversity of Roughly Balanced Bagging.

To evaluate diversity we calculated the disagreement measure [11]. For a pair of classifiers it is defined as a ratio of the number of examples on which both classifiers make different predictions to the number of all classified examples. This measure is calculated for each pair of component classifiers. Then the global, averaged disagreement D of an ensemble is averaged over all pairs of classifiers. The larger its value is, the more diverse classifiers are [11]. We calculated the global average disagreement D for predictions in both classes and also for the minority class only (denoted as D_{min}). These values are presented in Table 2 - two first columns for RBBag ensemble and next columns refer to its extension discussed in Sec. 5 – both ensembles were constructed with 30 component J48 trees. As this table concerns further extension of RBBag for a higher number of attributes, the list of datasets is reduced.

Notice that values of disagreement measures are relatively low. For nearly all datasets they are between 0.1 and 0.3. The small diversity concerns both class predictions (D) and minority class (D_{min}), although D_{min} is usually lower than D. Similar low values occurred for the remaining datasets, not included in Table 2. We also checked that changing the number of component classifiers in RBBag did not influence values of the disagreement measures.

To sum up, the high accuracy of Roughly Balanced Bagging is not directly related to its higher diversity. We have also analysed predictions of particular pairs of classifiers and noticed that they quite often make the same decisions (most often correct ones).

4 Influence of the Type of Examples

According to Napierala and Stefanowski [13, 14] the data difficulty factors concerning distributions of imbalanced classes can be modeled by the following types of examples: *safe examples* (located in the homogeneous regions populated by examples from one class only); *borderline* (placed close to the decision boundary between classes); *rare examples* (isolated groups of few examples located deeper inside the opposite class), or *outliers*. Following the method introduced in [13] the type of example can be identified by analysing class labels of the k-nearest neighbours of this example. For instance, if $k = 5$, the type of the example is assigned in the following way [13, 14]: 5:0 or 4:1 – an example is labelled as safe example; 3:2 or 2:3 – borderline example; 1:4 – labelled as rare example; 0:5 – example is labelled as an outlier. This rule can be generalized for higher k values, however, results of recent experiments [14] show that they lead to a similar categorization of considered datasets. Therefore, in the following study we stay with $k = 5$.

Repeating conclusions from experimental studies [13, 14] the most of datasets considered in this paper contain rather a small number of safe examples from the minority class. The exceptions are two datasets composed of many safe examples: `new-thyroid`, and `car`. Many datasets such as `cleveland`, `balance-scale` or `solar-flare` do not contain any safe examples but many outliers and rare cases.

In the current experiments we identified a type of the testing example and recorded whether it was correctly classified or not. Additionally, we refer types of examples in both (minority and majority) classes to the relative margins of the RBBag predictions (these are presented as histograms of numbers of testing examples with a given value of the margins). In Fig 3 and 4 we present a representative results of RBBag and the standard bagging for `cleveland` dataset. Histograms for other datasets present similar observations.

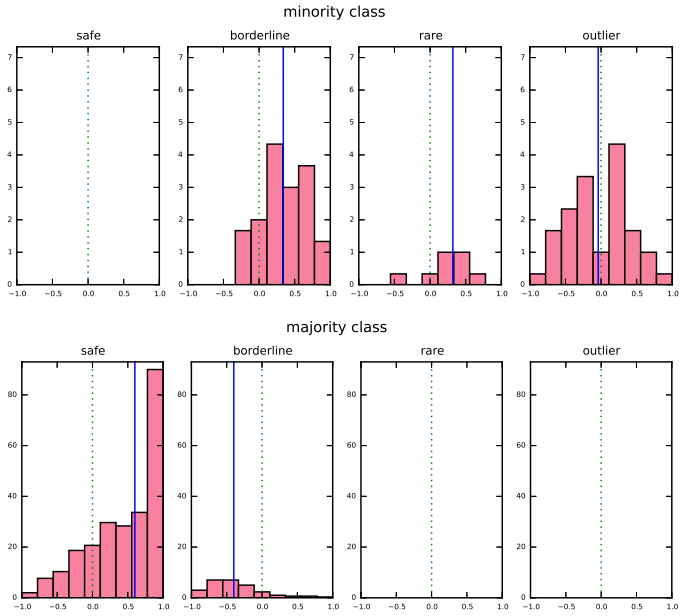


Fig. 3. Histogram of RBBag margins for `cleveland` dataset with respect to a class and a type of example. Blue vertical line shows the value of the margin's median.

Notice that RBBag quite well recognizes the borderline examples from the minority class. Rare minority examples are more difficult, however, on average RBBag can still recognize many of them. It classifies them much better than the standard bagging. Outliers are the most difficult, but RBBag classifies correctly some of them and again this is the main difference to standard bagging and other its over-sampling extensions evaluated in [3]. The similar tendency is observed for other unsafe datasets which are not visualized due to page limits. If the

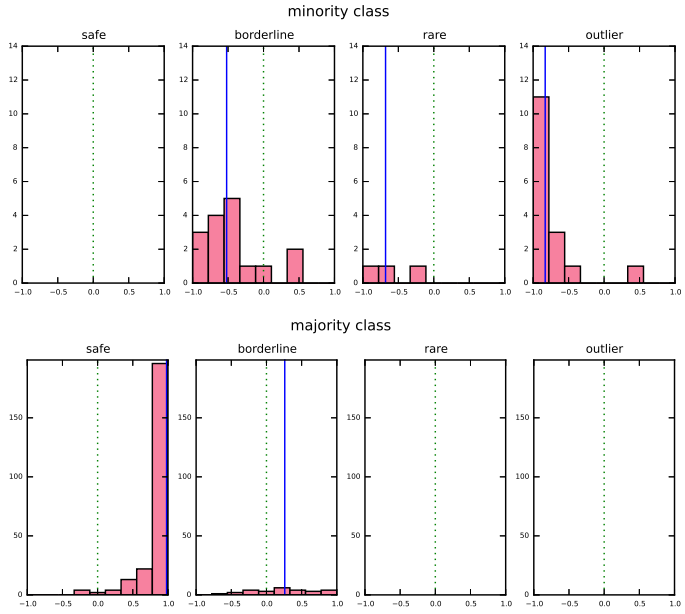


Fig. 4. Histogram of standard bagging margins for `cleveland` dataset with respect to a class and a type of example. Blue vertical line shows the value of the margin's median.

dataset contains some safe minority examples, nearly all of them are correctly classified with high margins.

On the other hand, for the majority class, one can notice that RBBag correctly classifies most of safe examples while facing difficulties with borderline ones. It also holds for other non-visualized datasests (where the margin's median for borderline majority examples is always worse than the median for borderline minority examples). The majority class does not contain any rare or outlying examples for nearly all considered datasets. For few exceptions, `pima`. `breast-cancer` or `cmc`, these rare majority examples are misclassified with the high negative margin.

In conclusion, we can hypothesize that Roughly Balanced Bagging improves recognition of unsafe minority examples, but at the cost of worse dealing with unsafe majority examples. However, as the number of unsafe examples is relatively small in the majority class, the final performance of RBBag (e.g., averaged by G-mean) is improved.

5 Applying Random Selection of Attributes

Although Roughly Balanced Bagging performs quite well, it can still be improved. Here, we have focused on modifications of constructing bootstrap samples. Observations of rather limited diversity of RBBag components have led us

to considering inspirations from earlier research on applying random attribute selection while constructing component classifiers. Recall that Ho introduced in [7] *Random Subspace* method (RSM) for highly dimensional data, where in each iteration of constructing the ensemble a subset of all available attributes is randomly drawn and a component classifier is built using only this subset. Then, Breiman combined bootstrap sampling with random selection of attributes in nodes of trees inside the Random Forest ensemble. Recent experiments of [12] also demonstrated that combing re-sampling with Random Forests helps for class imbalance. However, we are more interested in adapting Random Subspace into the context of Roughly Balanced Bagging as it is a classifier independent strategy. To best of our knowledge it has not been considered for RBBag yet. In the only related work [8] authors successfully applied this method to SMOTE based oversampling ensemble.

In our extension of RBBag, after sampling each bootstrap we randomly select f attributes from the set of all attributes. Subsequently, we train base classifier on a sample from which we removed not selected ones. We denote this extension as RBBag+RSM.

Since RSM is a method designed for high-dimensional data, we have chosen to our experiments only these datasets from earlier phases of experiments, which contain more then 11 attributes. As this condition holds for 9 datasets only, we added 4 new, high-dimensional imbalanced datasets from UCI repository. Finally, in this experiment we examine 13 following datasets: `abdominal-pain` (13 attributes), `cleveland` (13), `credit-g` (20), `dermatology` (35), `hepatitis` (19), `ionosphere` (34), `satimage` (37), `scrotal-pain` (13), `segment` (20), `seismic-bumps` (19), `solar-flare` (12), `vehicle` (18) and `vowel` (14).

We tested with J48 decision tree (without pruning) and SVM as base classifiers. Following the literature review, we considered setting f parameter to $\lceil\sqrt{F}\rceil$, $\lceil\log_2 F+1\rceil$ and $\lceil{}^1/_2 F\rceil$, where F is the total number of attributes in the dataset. Due to space limit we present results only for J48 decision trees and $f = \lceil\sqrt{F}\rceil$, since this parameter setting gives, on average, the highest increments.

We consider RBBag ensemble containing 30 component classifiers to be consistent with earlier experiments, in particular on diversity. However, following earlier observations, as e.g. [7], that randomization of attributes should increase the variance of bootstrap samples, we compare RBBag against the new proposed RBBag+RSM ensemble having also larger sizes (besides 30, also 50 and 70 components).

The values of G-mean and sensitivity are presented in Table 1. One can notice increases of both measures, in particular RBBag+RSM with more trees. For instance, the increase on sensitivity (`abdominal-pain`, `hepatitis` – above 6%) and G-mean (`abdominal-pain`, `hepatitis`, `scrotal-pain`, `seismic- bumps` – above 3%). We performed the paired Wilcoxon test to compare RBBag+RSM against RBBag. With the confidence $\alpha = 0.05$, RBBag+RSM is better on G-mean for 50 ($p = 0.007$) and 70 ($p = 0.003$) trees and nearly for 30 trees ($p = 0.054$). Similar results of this text hold for the sensitivity measure. Thus, it is better to construct RBBag+RSM with more trees than its RBBag equivalent.

	Sensitivity				G-mean			
	RBBag	RBBag+RSM			RBBag	RBBag+RSM		
Dataset	30	30	50	70	30	30	50	70
abdominal-pain	0.7955	0.8523	0.8623	0.8563	0.8077	0.8336	0.8411	0.8358
cleveland	0.7067	0.6800	0.7117	0.7567	0.7161	0.6938	0.7197	0.7410
credit-g	0.6610	0.6493	0.6407	0.6540	0.6735	0.6930	0.6923	0.7007
dermatology	0.9900	1.0000	1.0000	1.0000	0.9868	0.9986	1.0000	1.0000
hepatitis	0.7500	0.8200	0.8267	0.8267	0.7663	0.8131	0.8113	0.8029
ionosphere	0.8553	0.8660	0.8737	0.8796	0.9063	0.9068	0.9104	0.9152
satimage	0.8690	0.8738	0.8720	0.8777	0.8727	0.8677	0.8678	0.8698
scrotal-pain	0.7400	0.7467	0.7560	0.7453	0.7484	0.7869	0.7846	0.7884
segment	0.9863	0.9918	0.9933	0.9930	0.9892	0.9945	0.9955	0.9953
seismic-bumps	0.6312	0.6624	0.6629	0.6612	0.6824	0.7103	0.7153	0.7124
solar-flare	0.8690	0.8450	0.8670	0.8670	0.8499	0.8351	0.8437	0.8458
vehicle	0.9688	0.9990	0.9990	0.9990	0.9525	0.9590	0.9588	0.9599
vowel	0.9667	0.9911	0.9911	0.9900	0.9623	0.9751	0.9766	0.9789

Table 1. Sensitivity and G-mean for Roughly Balanced Bagging (RBBag) and its modification by random attribute selection (RBBag+RSM).

Additionally we calculated the disagreement measure for all examples (D) and also the minority class (D_{min}). The values presented in Table 2 are calculated for 30 trees. For reader convenience we present our results together with difference of disagreement between RBBag+RSM and original RBBag.

One can notice that Random Subspace method resulted in an increase of disagreement on almost all data sets (except `seismic-bumps`). Interestingly, despite a decline of the disagreement measure on this dataset we observed improvement on both G-mean and sensitivity.

	RBBag		RBBag+RSM		Difference	
Dataset	D	D_{min}	D	D_{min}	D	D_{min}
abdominal-pain	0.1564	0.1310	0.2995	0.2580	0.1431	0.1269
cleveland	0.2807	0.2470	0.3506	0.3050	0.0700	0.0581
dermatology	0.0211	0.0162	0.1815	0.1384	0.1604	0.1222
credit-g	0.2648	0.2279	0.4075	0.3951	0.1427	0.1672
hepatitis	0.2476	0.2127	0.3156	0.2915	0.0680	0.0788
ionosphere	0.0733	0.0909	0.1158	0.1650	0.0424	0.0741
satimage	0.1549	0.1160	0.1782	0.1448	0.0233	0.0288
scrotal-pain	0.1871	0.1670	0.3522	0.3139	0.1651	0.1469
semgent	0.0168	0.0106	0.0659	0.0293	0.0491	0.0187
seismic-bumps	0.2891	0.2373	0.2470	0.2383	−0.0421	0.0010
solar-flare	0.1062	0.0999	0.2362	0.2395	0.1300	0.1396
vehicle	0.0592	0.0509	0.1461	0.0972	0.0869	0.0463
vowel	0.0461	0.0251	0.2126	0.0825	0.1665	0.0574

Table 2. Disagreement measures, calculated for examples from both classes (D) and from the minority class only (D_{min}), for Roughly Balanced Bagging (RBBag) and its modification by random attribute selection (RBBag+RSM).

6 Discussion and Final Remarks

This study attempts to extend knowledge on properties of Roughly Balanced Bagging, which is one of the most accurate ensemble dedicated for class imbal-

ances. Our experiments show that it can be constructed with a relatively small number of component classifiers (approx. 15 ones). It is an interesting observation, as this ensemble may require a heavy under-sampling. One could expect that due to such strong changes inside distributions in bootstrap samples, their variance will be high, and the ensemble should reduce it by applying many components. However, the experimental results have showed that it is not a case. Moreover, this can be a promising indication for mining complex, larger data and for constructing this ensemble in an iterative way (starting from a smallest size and stepwise adding a new component while testing it with the extra validation set). According to other experiments the choice of the considered algorithms for learning component classifiers does not influence the final performance of RBBag

Another discovery is quite low diversity of RBBag. We have also confirmed it by calculating Q statistics diversity measure [11, 17]. Comparing it to earlier results [2] we argue that RBBag is less diversified than over-bagging or SMOTE-based bagging. On the other hand, RBBag is more accurate than these more diversified ensembles. We have also checked that its components are quite accurate and pairs of classifiers often make the same correct decisions. It may open another research on studying the trade off between accuracy and diversity of ensembles for imbalanced data.

Studying the local recognition of types of classified examples shows that RBBag improves classification of unsafe minority examples. Its power for dealing with borderline, rare and outlying examples distinguishes it from other ensembles. Here, we recall experiments from [3], which were focused on analysing distributions of example types in bootstraps. Their results revealed that several unsafe minority examples from the original data were changed by RBBag bootstrap sampling into safer ones which was not a case for other bagging extensions.

In this study we advocate for further modifications of bootstrap sampling. Our experiments have demonstrated that an integration of random selection of attributes improves the classification performance. However, other modifications could be still considered. In [3] we have already introduced Nearest Balanced Bagging which exploits information on types of minority examples and directs sampling towards the more unsafe examples. Although its experimental results are encouraging (for some datasets even better than RBBag) it generates bootstrap samples containing more minority examples than majority ones. Thus, it may be still shifted too much to improving sensitivity at the cost of removing too many majority examples. Recall that experiments from Sec. 4 have shown that RBBag also improves recognition of unsafe minority examples while worsening classification of borderline majority examples. Considering different types of examples from both classes while modifying bootstrap sampling in Roughly Balanced Bagging is still an open challenge.

Furthermore, a decomposition of classes into sub-concepts [9] could be considered. In [15] authors applied k-means clustering to stratify sampling majority examples inside their modifications of standard bagging. Looking for another semi-supervised clustering to better handle complex boundaries of data distributions could be yet another direction for future research.

Ack. The research was supported by NCN grant DEC-2013/11/B/ST6/00963.

References

1. Anyfantis, D., Karagiannopoulos, M., Kotsiantis, S., Pintelas, P. : Creating ensembles of classifiers by distributing an imbalance dataset to reach balance in each resulting training set. In Proc. of the IEEE DHMS Conf., (2008).
2. Błaszczyński, J., Stefanowski, J., Idkowiak L.: Extending bagging for imbalanced data. Proc. of the 8th CORES 2013, Springer Series on Advances in Intelligent Systems and Computing 226, 269–278 (2013).
3. Błaszczyński, J., Stefanowski, J.: Neighbourhood Sampling in Bagging for Imbalanced Data. Neurocomputing, vol. 150 A, 184-203 (2015).
4. Galar, M., Fernandez, A., Barrenechea, E., Bustince, H. Herrera, F.: A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 99, 1–22 (2011).
5. He, H., Yungian, Ma (eds): Imbalanced Learning. Foundations, Algorithms and Applications. IEEE - Wiley, (2013).
6. Hido S., Kashima H.: Roughly balanced bagging for imbalance data. In Proc. of the SIAM Int. Conference on Data Mining, 143-152 (2008) - an extended version in Statistical Analysis and Data Mining, vol. 2 (5-6), 412–426 (2009).
7. Ho, T.: The random subspace method for constructing decision forests. Pattern Analysis and Machine Intelligence 20(8), 832-844 (1998).
8. Hoens, T., Chawla, N.: Generating Diverse Ensembles to Counter the Problem of Class Imbalance. In Proc. PAKDD 2010, 488-499 (2010).
9. Jo, T., Japkowicz, N.: Class Imbalances versus small disjuncts. ACM SIGKDD Explorations Newsletter, vol. 6 (1), 40–49 (2004).
10. Khoshgoftaar T., Van Hulse J., Napolitano A.: Comparing boosting and bagging techniques with noisy and imbalanced data. IEEE Transactions on Systems, Man, and Cybernetics–Part A, 41 (3), 552–568 (2011).
11. Kuncheva, L.: Combining Pattern Classifiers. Methods and Algorithms. Wiley (2d Edition) (2014).
12. Liu A., Zhu Zh: Ensemble methods for class imbalance learning. In He, H., Yungian Ma. (eds): Imbalanced Learning. Foundations, Algorithms and Applications. Wiley, 61-82 (2013).
13. Napierala, K., Stefanowski, J.: The influence of minority class distribution on learning from imbalance data. In. Proc. 7th Conf. HAIS 2012, LNAI vol. 7209, Springer, 139-150 (2012).
14. Napierala, K., Stefanowski, J.: Types of Minority Class Examples and Their Influence on Learning Classifiers from Imbalanced Data. Journal of Intelligent Information Systems, on-line access DOI: 10.1007/s10844-015-0368-1 (2015).
15. Parinaz, S., Victor, H., Matwin, S.: Learning from Imbalanced Data Using Ensemble Methods and Cluster-based Undersampling. In Post- Proc. 3rd Workshop New Frontiers of Mining Complex Patterns at ECML-PKDD 2014, Nancy, LNAI vol. 8983, Springer, 69 – 86 (2015).
16. Wallace, B., Small, K., Brodley, C., Trikalinos, T.: Class Imbalance, Redux. Proc. 11th IEEE International Conference on Data Mining, 754 – 763 (2011).
17. Wang, S., Yao, T.: Diversity analysis on imbalanced data sets by using ensemble models. In Proc. IEEE Symp. Comput. Intell. Data Mining, 324-331 (2009).

Classifying traces of event logs on the basis of security risks

Bettina Fazzinga[1], Sergio Flesca[2], Filippo Furfaro[2], and Luigi Pontieri[1]

[1] ICAR-CNR, Italy – {fazzinga, pontieri}@icar.cnr.it
[2] DIMES, University of Calabria, Italy – {flesca, furfaro}@dimes.unical.it

Abstract. In the context of security risk analysis, we address the problem of classifying log traces describing business process executions. Specifically, on the basis of some (possibly incomplete) knowledge of the process structures and of the patterns representing unsecure behaviors, we classify each trace as instance of some process and/or as potential security breach. This classification is addressed in the challenging setting where each event has not a unique interpretation in terms of the activity that has generated it, but it can correspond to more activities. In our framework, the event/activity mapping is encoded probabilistically, and the models describing the processes and the security breaches are expressed in terms of precedence/causality rules over the activities. Each trace is classified on the basis of the conformance of its possible interpretations, generated by a Monte Carlo mechanism, to the security-breach models and/or the process models. The framework has been experimentally proved to be efficient and effective.

1 Introduction

Despite the adoption of automated control, monitoring, and tracing infrastructures, and of and access/usage control mechanisms, business processes are continuously exposed to security threats (e.g., financial frauds, data leakage, system faults, regulatory non-compliance). In fact, security breaches tend to emerge frequently in real-world processes, and this may severely undermine the achievement of business goals, or severely damage the organization, as witnessed by recent scandals like the 2011 UBS rogue trader scandal. This clearly calls for analyzing the actual behavior of process instances, exploiting the execution traces that are typically registered by process enactment systems. However, as discussed in [5], due the lack of tools tailored at business process logs, current auditing practices often rely on manual inspections and/or on sampling.

This explains the recent interest [4, 5, 9, 13] towards exploiting process mining techniques (such as workflow induction [1], compliance [3] and conformance [15] checking) for carrying out security-oriented analyses over business process logs. However, these techniques found on the assumption that each step in a log trace unambiguously refers to one of the activities that compose the "high-level" process models that security analysts and business users have in mind. Unfortunately, many enactment and tracing systems work at a lower abstraction level: each event in a trace represents the execution of a fine-grain operation, with no clear mapping to a unique "high-level" activity.

In this work, we face the problem of classifying business process traces in the context of security risk analysis: based on some knowledge on the structures of the processes

and of the patterns representing undesired/risky behaviors (encoded into *process models* and *security-breach models*, respectively), we aim at classifying each trace as instance of some process and/or as potential security breach. In particular, the problem is addressed in the above-introduced challenging setting: the models feature high-level activities (and encode precedence/causality constraints over the activities, like in declarative process modeling frameworks [2, 16]); conversely, each trace is a sequence of low-level events, and can be regarded as the execution of one among many possible activities. Consequently, multiple interpretation may exist for each trace τ, in terms of sequences of activities that might have generated (the sequence of events composing) τ.

This means that performing a security analysis in the considered setting requires addressing the uncertainty inherent to interpreting traces as sequences of activity executions. In order to deal with this uncertainty, we encode the mapping from events to activities by way of probability distributions, and address the problem of classifying each trace based on the conformance of its possible interpretations to the security-breach models and/or to the process models. In this regard, we address two possible scenarios: the *open world* scenario, where the set of models in input is possibly incomplete (i.e., some traces might have been produced by unknown process models, and may follow unknown kinds of breaches); and the *closed world* scenario, where a model is given for every possible business processes and security breaches. In the latter scenario, each trace is known to be aligned with at least one of the process models, and it can be (probabilistically) assigned to two classes only ("breaches" vs. "non-breaches"). Two similar further classes are to be considered in the open world scenario, in order to encompass the event that the trace does not comply with none of the given process models. Due to the one-to-many mapping from events to activities, estimating the probability that a trace τ belongs to one of these classes would require exploring a possibly very large (combinatorial) number of possible interpretations of τ. Since this may be unfeasible in many real-world scenarios (as shown in the experimentation section), we propose to adopt an ad-hoc Monte Carlo sampling method, allowing to efficiently obtain good estimates of the class-membership probabilities (based on a reduced number of interpretations). The proposed framework has been experimentally validated, and proved to be efficient and effective.

Related work. Our model-driven detection of security breaches shares some connection with the activity detection and situation awareness [10] problems, and with some intrusion/threat detection approaches relying on event-based attack models (e.g., attack tree/graphs [14]. Notably, the system presented in [10] allows the analyst to monitor multiple FSMs modeling malicious/undesired behavior, by maintaining multiple "interpretation" hypotheses concurrently. However, all of the above approaches are process-unaware, as they do not take into account background information on process structure.

Process-awareness is a central issue instead in the areas of Business Process Management (BPM) and Business Process Intelligence (BPI). Several efforts were done in the last decade in order to integrate risk-oriented analysis mechanisms into the design, monitoring and ex-post analysis of business processes (see [18] for a recent survey on this topic). As to security-oriented risks, several log-centered approaches were proposed recently [13, 9, 5, 4] that leverage different kinds of techniques developed in the Process Mining community. In particular, the usage of workflow induction [1] (resp., of com-

pliance checking [3] and conformance checking [15]) techniques was proposed in [5] (resp., in [4]) as a way to carry out security-oriented analyses over large amounts of log data. The recognition of high-risk process instances was also faced by using a model of forbidden/undesired behavior [17], and evaluating whether a trace is an instance of it. However, to the best of our knowledge, there is no solution for the detection of security breaches that can apply to low-level logs.

Notably our idea of exploiting constraint-based process models as a form of background knowledge (especially in the "closed-world assumption" setting) is novel in the context of security-breach detection. The usage of a-priori inter-activity (precedence) constraints in a BPI setting has only been considered in the discovery of workflow models, in order to prune the search space [12] and obtain higher quality models.

Finally, the problem of mapping log events to pre-defined process activities was faced in [7, 8], where semi-automated approaches are devised for converting each original log trace into a sequence of the given activities. However, none of these approaches can hence solve the specific problem stated in this work, where a log trace is to be contrasted to multiple models (representing process behaviors and security breach patterns), in order to possibly interpret it as an instance of each of them.

2 Preliminaries

Logs, traces, processes, activities and events. A log is a set of *traces*. Each trace Φ describes a process instance at the abstraction level of basic *events*, each generated by the execution of an activity. That is, an instance w of a *process* W consists of a sequence $a_1, \ldots, a_n$ of *activity* instances; in turn, each activity instance a_i generates an event e_i; hence, the trace Φ describing w consists in the sequence $e_1, \ldots, e_n$. For any event e_i occurring in a trace, we assume that the starting time point of its execution is represented in the log, and denote it as $e_i.t_s$. We also assume that any activity instance a_i "inherits" the starting time point from the generated event e_i, and denote it as $a_i.t_s$.

In the following, we assume given the sets $\mathcal{W}$, $\mathcal{A}$ of (types of) processes and activities, respectively, and the set $\mathcal{E}$ of the events that can occur in the log. We denote the elements of $\mathcal{W}$ and $\mathcal{A}$ with upper-case alphabetical symbols (such as W, A). The instances of processes and activities, as well as the events, will be denoted with lower-case symbols (such as w, a, e).

Mapping events onto activities. Typically, the correspondence between activities and events is *one to many*, that is: 1) for every activity $A \in \mathcal{A}$, all the executions of A generate the same event e; 2) conversely, an occurrence of e in some trace of the log cannot be univocally interpreted as the result of an execution of A, since there can be another activity $B \in \mathcal{A}$ whose execution may have generated e. We assume that the mapping of each event $e \in \mathcal{E}$ onto the activities is probabilistically modeled by a pdf $p_e(A)$, where A is a random variable ranging over $\mathcal{A}$. Basically, $p_e(A)$ encodes the probability that a generic occurrence of the event e has been caused by an execution of activity A. It is worth noting that $p_e(A)$ can be obtained by encoding the knowledge of domain experts, or (as done in our experiments) using a training set of traces from which statistics can be evaluated on how frequently the event e is generated by an instance of A, for each $e \in \mathcal{E}$ and $A \in \mathcal{A}$.

Interpretations and their probabilities. An activity instance a of A is said to be an *interpretation* of e (or, equivalently, *compatible* with e) if $p_e(A) > 0$ (meaning that e can be reasonably viewed as the result of executing A). Given a trace $\Phi = e_1 \dots e_n$, we call *interpretation* of Φ a sequence of activity instances $a_1 \dots a_n$ such that, $\forall i \in [1..n]$, a_i is an interpretation of e_i. The set of interpretations of Φ will be denoted as $\mathcal{I}(\Phi)$.

$p_e(A)$ can be used as the core of a naive mechanism for defining a pdf over $\mathcal{I}(\Phi)$: assuming independence between the events, each $I = a_1 \dots a_n$ in $\mathcal{I}(\Phi)$ can be assigned $p(I) = \Pi_1^n p_{e_i}(A_i)$ as the probability of being the sequence of events that generated Φ.

Example 1. Let $\mathcal{A} = \{A, B, C\}$, $\mathcal{E} = \{e_1, e_2\}$, and consider the log $\mathcal{L} = \{\Phi\}$, where $\Phi = e_1 e_2$. Assume that $p_{e_1}(A) = p_{e_1}(B) = 0.5$ (meaning that the executions of activities A and B are equi-probable causes of event e_1), and that $p_{e_2}(C) = 1$ (meaning that every occurrence of event e_2 can be generated only by an execution of C). We denote as a, b, c the generic instances of A, B and C, respectively. Given this, trace Φ can be interpreted as either the sequence of activity instances $I_1 = a\,c$ (with probability $p_{e_1}(A) \cdot p_{e_2}(C) = 0.5$) or $I_2 = b\,c$ (with probability $p_{e_1}(B) \cdot p_{e_2}(C) = 0.5$). □

Process models and security-breach models. We assume that some knowledge of the structure of every process $W \in \mathcal{W}$ can be encoded in terms of a set $W.\mathcal{IC}$ of *composition rules*, restricting the sequences of activity instances that are allowed to be executed within W. Basically, a composition rule has one of the following forms: 1) $A \Rightarrow_T B$; 2) $A \Rightarrow_T \neg B$; 3) $A \Leftarrow_T B$; 4) $\neg A \Leftarrow_T B$; where A, $B \in \mathcal{A}$, while T is of the form '$\leq c$', where c is a constant. Herein, $A \Rightarrow_T B$ (resp., $A \Rightarrow_T \neg B$) imposes that, within every instance of W, the execution of any instance a of A must (resp., must not) be *followed* by the execution of an instance b of B such that the width of the interval between $a.t_s$ and $b.t_s$ satisfies T. The semantics of the forms $A \Leftarrow_T B$ and $\neg A \Leftarrow_T B$ is analogous: they have to be read from the right to the left, replacing the word "*followed*" with "*preceded*" in the above definition. Omitting T is the same as specifying $T =$'$\leq \infty$'. Special cases of the rules of the form $A \Rightarrow B$ and $A \Rightarrow \neg B$ are the rules *true* $\Rightarrow B$ and *true* $\Rightarrow \neg B$ meaning that the activity B must (resp., must not) occur within the executions of the process, respectively.

Example 2. Let W be a process with $W.\mathcal{IC} = \{A \Rightarrow \neg B; C \Leftarrow A\}$. Then, the sequence $a\,b\,c$, whose elements are instances of the activities A, B, C, cannot be an instance of W, since it violates both the composition rules. On the contrary, the sequence $c\,a$ conforms to $W.\mathcal{IC}$, thus it can be viewed as an instance of W. □

We also assume that some knowledge of security risks is encoded in terms of *security-breach models*. A security-breach model is a set of composition rules describing causality and precedence relationships between activity executions that describe risky situations. The composition rules have the same syntax as those used for the process models. In the following, we denote as $\mathcal{SBM}$ the set of security-breach models, and, for each security-breach model $SBM \in \mathcal{SBM}$, we denote the set of composition rules associated with SBM as $SBM.\mathcal{IC}$.

Example 3. Let SBM be a security breach model with $SBM.\mathcal{IC} = \{\neg A \Leftarrow_{<12hr} B\}$, where A is the activity "*Detection of critical trouble*" and B is "*Communication with*

the customer via private channel". The composition rule of SBM means that the case of a customer who has been contacted via a private channel without the existence of a recently detected critical trouble is a security breach that should be looked into. □

Open world and closed world assumptions. We consider two scenarios, corresponding to different levels of completeness of the knowledge on the structure of the processes and of the patterns describing security breaches. The open world scenario is the case that this knowledge is incomplete: $\mathcal{W}$ (resp., $\mathcal{SBM}$) is a (possibly strict) subset of the set of models describing the structure of all the possible processes (resp., security breaches) that can occur in the log. On the contrary, the closed world scenario is the case that all the possible processes and security breaches conform to some model in $\mathcal{W}$ and $\mathcal{SBM}$, respectively. For instance, under the open world assumption, the meaning of $\mathcal{W} = \{W_1\}$ and $\mathcal{SBM} = \{SBM_1, SBM_2\}$ is: *the only processes and security risks for which a model is known are W_1 and SBM_1, SBM_2, respectively; however, the occurrence of process instances and security breaches conforming to none of these models cannot be excluded.* Under the closed world assumption, there cannot be process instances that do not conform to W_1, and there are no patterns characterizing security breaches that do not conform to either SBM_1 or SBM_2.

3 The classification problem and our approach for solving it

The problem addressed in this paper is that of classifying the log traces on the basis of the knowledge of the process and security-breach models. We define the classification problem considering the open and closed world scenarios separately.

Classifying traces under the open world assumption. In this scenario, any sequence of activity instances belongs to exactly one of the following classes (the set of these classes will be denoted as $\mathcal{C}(ow)$):

- *Aligned*: sequences conforming to at least one process model $W \in \mathcal{W}$, but to no security breach model $SBM \in \mathcal{SBM}$;
- *Breach*: sequences conforming to at least one $SBM \in \mathcal{SBM}$, but to no $W \in \mathcal{W}$;
- *Aligned&Breach*: sequences conforming to at least one $W \in \mathcal{W}$ and at least one $SBM \in \mathcal{SBM}$;
- *Unknown*: sequences conforming to no $W \in \mathcal{W}$ and no $SBM \in \mathcal{SBM}$.

Classifying a trace $\Phi = e_1 \dots e_n$ means marking it with the name of the class containing the interpretation $a_1 \dots a_n$ that generated $e_1 \dots e_n$. Since Φ has many possible interpretations, where each $I \in \mathcal{I}(\Phi)$ has probability $p(I)$ of being the actual "origin" of Φ, this problem can be re-written under a probabilistic standpoint: for each class $C \in \mathcal{C}(ow)$, evaluate the probability $p^{ow}(\Phi, C)$ that the sequence of activities that actually generated Φ is in C. This means evaluating, for each $C \in \mathcal{C}(ow)$, the probability:

$$p^{ow}(\Phi, C) = \sum_{I \in (\mathcal{I}(\Phi) \cap C)} p(I). \tag{1}$$

Observe that, in the open world scenario, the independence assumption on which the definitions of $p(I)$ and $p^{ow}(\Phi, C)$ are based is reasonable, since no structure for the

processes can be excluded: this backs assuming no correlation between the activities.

Classifying traces under the closed world assumption. The closed world assumption describes the case that the possible structures of the processes and security breaches are exhaustively described by the models in $\mathcal{W}$ and $\mathcal{SBM}$. Hence, only the classes *Aligned* and *Aligned&Breach* must be considered to mark every trace Φ in the log. The set consisting of these two classes will be denoted as $\mathcal{C}(cw)$. As regards the probability $p^{cw}(\Phi, C)$ that, under the closed world assumption, the actual interpretation of Φ belongs to the class C (where $C \in \mathcal{C}(cw)$), its definition is more complex than its "open-world" counterpart $p^{ow}(\Phi, C)$. We cannot define $p^{cw}(\Phi, C) = p^{ow}(\Phi, C) = \sum_{I \in \mathcal{I}(\Phi) \cap C} p(I)$ since the definition of $p(I)$ relies on the independence assumption, that is no longer valid in this scenario. Indeed, the fact that only the interpretations satisfying some process model must be considered for the classification purposes means that there are correlations between the activity executions: hence, the events cannot be considered independent from one another. A reasonable way to define $p(\Phi, C)$ is that of applying the probabilistic conditioning paradigm, that is a way to revise the pdf that would hold under independence assumption by a-posteriori enforcing some correlations. In our case, this means defining $p^{cw}(\Phi, C) = p^{ow}(\Phi, C|\mathcal{W}.\mathcal{IC})$, where the right-hand side of this expression is the result of conditioning the probability $p^{ow}(\Phi, C)$ to the event that the constraints expressed by the composition rules of at least one process model in $\mathcal{W}$ are satisfied. This means that, for each $C \in \mathcal{C}(cw)$:

$$p^{cw}(\Phi, C) = \frac{p^{ow}(\Phi, C)}{p^{ow}(\Phi, \textit{Aligned}) + p^{ow}(\Phi, \textit{Aligned\&Breach})}. \tag{2}$$

An example of classification. The following example shows an example of computation of the probabilities $p^{ow}(\Phi, C)$ (for each $C \in \mathcal{C}(ow)$) and $p^{cw}(\Phi, C)$ (for each $C \in \mathcal{C}(cw)$), and discusses the meaning of the classification.

Example 4. Consider the case described in Example 1 of the trace $\Phi = e_1 e_2$, having two interpretations: $I_1 = a\,c$ and $I_2 = b\,c$, with $p(I_1) = p(I_2) = 0.5$. Assume that $\mathcal{W} = \{W\}$ and $\mathcal{SBM} = \{SBM\}$, where $W.\mathcal{IC} = \{A \Leftarrow C; B \Rightarrow C\}$ and $SBM.\mathcal{IC} = \{B \Leftarrow C\}$. It is easy to see that I_1 conforms to $W.\mathcal{IC}$, but not to $SBM.\mathcal{IC}$. Moreover, I_2 does not conform to $W.\mathcal{IC}$, but conforms to $SBM.\mathcal{IC}$.

Under the open world assumption, the classes to be considered are: $\mathcal{C}(ow) = \{$*Aligned*, *Breach*, *Aligned&Breach*, *Unknown* $\}$. Then, the classification of Φ consists in the following probability assignment: $p^{ow}(\Phi,$ *Aligned*$) = p(I_1) = 0.5$; $p^{ow}(\Phi,$ *Breach*$) = p(I_2) = 0.5$; $p^{ow}(\Phi,$ *Aligned&Breach*$) = p^{ow}(\Phi,$ *Unknown*$) = 0$.

On the contrary, under the closed world assumption, the classes to be considered are: $\mathcal{C}(cw) = \{$*Aligned*, *Aligned&Breach* $\}$, and the classification of Φ is as follows: $p^{cw}(\Phi,$ *Aligned*$) = p(I_1)/p(I_1) = 1$; $p^{cw}(\Phi,$ *Aligned&Breach*$) = 0$.

The classification under the closed world assumption can be read as follows: with probability 1, Φ is consistent with the model of the process, and contains no security breach. Analogously, the classification under the open world assumption can be read as follows: with probability 0.5, the trace Φ is consistent with a known process model and does not contain any known security breach; with probability 0.5, Φ does not conform to any known process model but conforms to a security breach model. □

3.1 The challenges of evaluating a classification and our solution

Efficiently providing the classification of the traces in a log is a hard problem, independently from the fact that the open world (ow) or the closed world (cw) assumption is made. A naive approach for classifying a trace Φ is the following: 1) generate the set $\mathcal{I}(\Phi)$ of all of the interpretations of Φ; 2) for each $I \in \mathcal{I}(\Phi)$, check whether I conforms to some model in $\mathcal{W} \cup \mathcal{SBM}$; 3) depending on the models to which I conforms and on the assumption, mark I with the proper class in $\mathcal{C}(\textit{assumption})$; 4) for every class $C \in \mathcal{C}(\textit{assumption})$, compute $p^{\textit{assumption}}(\Phi, C)$ using equations (1) and (2).

Unfortunately, this naive approach is infeasible, as the interpretations to be considered may be too many. For instance, consider a trace Φ of length 40. If, on average, each event has two interpretations, then $|\mathcal{I}(\Phi)| = 2^{40} = 10^{12}$ interpretations for Φ must be considered. As we will show experimentally (see Section 5), deciding the class of all these interpretations typically requires very long waits.

In this paper, we investigate the possibility of using a Monte Carlo approach for evaluating the classification of each log trace Φ. Basically, we use a Monte Carlo sampler over $\mathcal{I}(\Phi)$, that randomly generates a new interpretation in $\mathcal{I}(\Phi)$ until the probabilities of the classes (evaluated only on the basis of the sample set generated so far, instead of the whole $\mathcal{I}(\Phi)$) converge to the actual probability values (the convergence is checked according to a given confidence level). We show that our approach is efficient under both the open and closed world assumptions, and feasible even in the cases where the exhaustive approach requires too much time.

4 The Monte Carlo classification algorithm

In this section, we describe our Monte-Carlo simulation approach and its implementation (Algorithm 1). Algorithm 1 takes as input the trace Φ, the sets of process and security breach models $\mathcal{W}$ and $\mathcal{SBM}$, the set P of pdfs $p_e(A)$, the assumption under which the classification must be evaluated (that is, either ow or cw), and two parameters defining the desired guarantee on the accuracy of the Monte Carlo estimation: an error level ϵ and a confidence level $1 - \alpha$. The output of Algorithm 1 is an estimate of the classification of Φ under the specified assumption: that is, Algorithm 1 returns an estimate $\tilde{p}^{\textit{assumption}}(\Phi, C)$ of $p^{\textit{assumption}}(\Phi, C)$ for each $C \in \mathcal{C}(\textit{assumption})$. In particular, it is guaranteed that the actual probability $p^{\textit{assumption}}(\Phi, C)$ is in the interval $\tilde{p}^{\textit{assumption}}(\Phi, C) \pm \epsilon$ with confidence level $1 - \alpha$.

In brief, Algorithm 1 samples the set $\mathcal{I}(\Phi)$ of interpretations and determines the class in $\{$*Aligned*, *Breach*, *Aligned&Breach*, *Unknown*$\}$ into which each sample I falls. At the end of the sampling process, for each $c \in \mathcal{C}(\textit{assumption})$, it returns as $\tilde{p}^{\textit{assumption}}(\Phi, C)$ the fraction of the samples belonging to class C. In more detail, Algorithm 1 works as follows. At line 1, it initializes the four variables n_A, n_B, n_{AB} and n_U used to store the number of sampled interpretations falling in the class *Aligned*, *Breach*, *Aligned&Breach*, and *Unknown*, respectively. The loop from line 2 to line 13 represents the core of the sampling process. At each iteration of this loop, an interpretation I is generated (lines 3-4), by randomly choosing one of the candidates activities for each event of the trace (line 4). The loop from line 6 to line 8 scans the process models in $\mathcal{W}$:

Algorithm 1 The interpretation algorithm

Input: A trace Φ, a set $\mathcal{W}$ of process models, a set $\mathcal{SBM}$ of security breach models, a set P containing a pdf $p_e(A)$ for every event $e \in \mathcal{E}$, a parameter *assumption* $\in \{ow, cw\}$, an error level ϵ, a confidence level $1-\alpha$.

Output: $\tilde{p}^{ow}(\Phi, C)$ for each C in {*Aligned, Breach, Aligned&Breach, Unknown*}, in the case *assumption=ow*; $\tilde{p}^{cw}(\Phi, C)$ for each C in {*Aligned, Aligned&Breach*}, otherwise.

1: $n_A = 0, n_B, n_{AB} = 0, n_U = 0, I = \perp$
2: **repeat**
3: **for all** $e_i \in \Phi$ **do**
4: $I[i]$ =*chooseActivity*(e_i)
5: *foundAlignment*=`false`, *foundBreach*=`false`
6: **for all** $W \in \mathcal{W}$ **do**
7: **if** *checkModel*$(I, W.IC, P)$ **then**
8: *foundAlignment*=`true`; `break`
9: **for all** $SBM \in \mathcal{SBM}$ **do**
10: **if** *checkModel*$(I, SBM.IC, P)$ **then**
11: *foundBreach*=`true`; `break`
12: *Update* n_A, n_B, n_{AB}, n_U *according to foundAlignment and foundBreach*
13: **until** *errorGuarantee*$(n_A, n_B, n_{AB}, n_U$, *assumption*, $\epsilon, 1-\alpha)$
14: **if** *assumption=ow* **then**
15: $n = n_A + n_B + n_{AB} + n_U$
16: $\tilde{p}^{ow}(\Phi$, *Aligned*$) = \frac{n_A}{n}$, $\tilde{p}^{ow}(\Phi$, *Breach*$) = \frac{n_B}{n}$, $\tilde{p}^{ow}(\Phi$, *Aligned&Breach*$) = \frac{n_{AB}}{n}$, $\tilde{p}^{ow}(\Phi$, *Unknown*$) = \frac{n_U}{n}$
17: **return** $\tilde{p}^{ow}(\Phi,$ *Aligned*$), \tilde{p}^{ow}(\Phi,$ *Breach*$), \tilde{p}^{ow}(\Phi,$ *Aligned&Breach*$), \tilde{p}^{ow}(\Phi,$ *Unknown*$)$
18: $n = n_A + n_{AB}$
19: $\tilde{p}^{cw}(\Phi$, *Aligned*$) = \frac{n_A}{n}$, $\tilde{p}^{cw}(\Phi$, *Aligned&Breach*$) = \frac{n_{AB}}{n}$
20: **return** $\tilde{p}^{cw}(\Phi$, *Aligned*$), \tilde{p}^{cw}(\Phi$, *Aligned&Breach*$)$

if one of the process model of $\mathcal{W}$ is found such that I satisfies all the constraints in it, a boolean variable *foundAlignment* becomes `true` and the loop terminates. Analogously, the loop from line 9 to line 11 scans the security breach models in $\mathcal{SBM}$ and early terminates if a model is found such that I satisfies all the constraints in it. In that case, a boolean variable *foundBreach* becomes `true`. After the execution of the two loops scanning the process and the security breach models, respectively, *foundAlignment* and *foundBreach* are used to determine the class containing I and update n_A, n_B, n_{AB} and n_U accordingly (lines 12): if both (resp., none of the) variables *foundAlignment* and *foundBreach* are `true`, I falls into *Aligned&Breach* (resp., *Unknown*), thus n_{AB} (resp., n_U) is incremented. In the case that only *foundAlignment* (resp., *foundBreach*) is `true`, I falls into *Aligned* (resp., *Breach*), thus n_A (resp., n_B) is incremented.

The generation of samples is halted by function *errorGuarantee*, that exploits the Agresti-Coull interval [6] to detect if, for each $C \in \mathcal{C}$(*assumption*) the estimate $\tilde{p}^{assumption}(\Phi, C)$ of $p^{assumption}(\Phi, C)$ obtained with the samples collected so far lies in the interval $\tilde{p}^{assumption}(\Phi, C) \pm \epsilon$ with confidence level $1-\alpha$. We recall that, according to [6], the error of the estimate $\tilde{p}$ of the target probability p obtained after n samples is guaranteed to be at most ϵ with confidence level $1-\alpha$ if $n > \overline{n} = \frac{z^2_{1-\alpha/2} \cdot \overline{p} \cdot (1-\overline{p})}{\epsilon^2} - z^2_{1-\alpha/2}$,

where $z_{1-\alpha/2}$ is the $1-\alpha/2$ quantile of the normal distribution, and $\overline{p} = \frac{n_x+(z^2_{1-\alpha/2})/2}{n+z^2_{1-\alpha/2}}$, where n_x is the number of successes in the n samples. Hence, since the error guarantee must be provided for every estimate returned by Algorithm 1, function *errorGuarantee* computes $|\mathcal{C}(\textit{assumption})|$ values of $\overline{n}$, one for each estimate $\tilde{p}^{\textit{assumption}}(\Phi, C)$ that has to be returned. Correspondingly, Algorithm 1 halts only if the number of generated samples is equal to or greater than all these values of $\overline{n}$ (line 13).

5 Experimental Validation

Hardware settings and dataset features. All the experiments were done on an Intel i7 CPU with 12GB RAM running Windows 8.1. We tested our framework over synthetic real-like data, generated according to the guidelines of the administrative units of a service agency (SA). In this scenario, a process instance is a collection of activities performed by the staff of the units in response to customers' requests. Examples of activities are the creation of a new folder, the preparation of new documents and their insertion into a folder, the updating of existing documents, contacting the customer, etc.. Folders are of different categories, and folders of the same category follow the same execution scheme, that describes a process. We were given a set $\mathcal{W}$ of 6 processes and their models in terms of precedence relationships (between activities), that were easily encoded into composition rules of the form used in our framework. The scenario is complex enough that the same activities were shared by different processes (on average, an activity occurs in one half of the processes), and the same event could be generated by different activities (on average, the same event occurs in 2.3 different activities).

We were also given the models of 8 different types of security breaches, and each model was translated into a set of composition rules. On average, each process was described by 10 composition rules, and each security breach by 5 composition rules.

Besides the composition rules, the service agency gave us a set of 100 real traces describing different process instances at the abstraction level of events, along with their actual interpretations (that is, the corresponding sequences of activities). We used these traces and their interpretations for generating both the set P of pdfs of the form $p_e(A)$ and the larger dataset used in the experiments. As regards P, it was obtained by extracting statistics on the actual correspondences events/activities occurring in the pairs trace/interpretation given by the service agency. As regards the dataset, starting from each interpretation I, we generated a set *perturb*(I) of 100 sequences of activity instances, by suitably perturbing I. Specifically, *perturb*(I) was initially assigned the set consisting of only I, and then the $(i+1)$-th sequence in *perturb*(I) was obtained from the i-th one by applying one perturbation, randomly chosen among: (a) replacing a randomly chosen interpretation step with a new instance of a randomly chosen activity having the same starting time; (b) switching a randomly chosen interpretation step with the subsequent one; (c) removing a randomly chosen interpretation step; (d) inserting a new instance of a randomly chosen activity into the sequence at a random position (the starting time of the new activity is randomly generated in the interval between the starting times of the previous and the subsequent activities). Before adding the perturbed sequence of activity instances to *perturb*(I) as the $(i+1)$-th element, we checked its consistency with the models of the processes in $\mathcal{W}$, and we discarded it in the case of

inconsistency with every model (in this case, a new perturbation was tried over the i-th interpretation). Finally, once the generation of every *perturb*(I) was finished, each I' in *perturb*(I) was translated into the corresponding trace (i.e., sequence of events), and this was put in the dataset. This way, we obtained a dataset consisting of 10^4 traces.

Term of comparison, and open and closed world scenarios. We compared our approach with the naive exhaustive approach described in section 3.1, denoted as EX in the following. The experiments under the open world assumption were performed by making $\mathcal{W}$ and $\mathcal{SBM}$ consist of one half of the process and security breach models provided by the service agency, thus simulating the case that $\mathcal{W}$ and $\mathcal{SBM}$ do not encode a complete knowledge of the possible processes and security breaches. Obviously, for the closed world assumption, we put into $\mathcal{W}$ and $\mathcal{SBM}$ all the process and security breach models provided by the service agency, respectively.

Results. We start with comparing our Monte Carlo based approach with EX in terms of efficiency. In what follows, the two variants of Algorithm 1, corresponding to adopting either the open or the closed world assumption, are referred to as MC-OW and MC-CW, respectively. We make no distinction between the behaviors of the exhaustive approach under the two assumptions since there is no difference in terms of efficiency: under both the open and closed world assumptions, EX performs the same number of iterations (as all the interpretations must be generated and classified according to the models). All the results presented in what follows were obtained by setting $\epsilon = 0.001$ and $1 - \alpha = 95\%$.

Fig. $1(a)$ shows the average execution times of the considered approaches vs. the trace length. Even if our dataset contains longer traces, this diagram reports only the results for the traces whose length is less than 30, since EX required too much time ($\geq$ 20min) to complete the classification over longer traces. The results for EX are represented by 3 distinct curves, obtained as follows. First, the set of traces was partitioned into 3 sets, denoted as $(2.0{-}2.2)$, $(2.2{-}2.4)$, $(2.4{-}2.6)$: a trace Φ of the dataset belongs to the set $(X - Y)$ if the average number *ActPerEv*(Φ) of activities that are possible interpretations for a step of Φ belongs to the interval $(X - Y)$. Then, for each $(X - Y)$, the curve EX$(X - Y)$ represents the average execution time of EX over the traces in $(X - Y)$. This distinction was not made for MC-OW and MC-CW, since, as expected, their execution times turned out to be insensitive to *ActPerEv*(Φ). From this diagram, it turns out that execution times for EX grow exponentially with the trace length (in fact, the shape of the curves of EX are linear in the presence of a logarithmic scale on the y-axis), and that also increasing *ActPerEv*(Φ) results in slowing down EX.

Fig. $1(b)$ considers also traces longer than 30 steps, and depicts the average running time of MC-OW and MC-CW vs. trace length (for what explained before, EX is not considered in this diagram). The diagram shows that average execution times grow with the trace length (this sensitiveness was hidden in Fig. $1(a)$ by the use of a log scale).

We also analyzed the effectiveness of our Monte Carlo approach in terms of accuracy of the estimated classification. For each trace Φ, we measured the error of the estimate returned by Algorithm 1 as the maximum difference between the probability associated with a class C in the actual classification (that is, the classification returned by EX) and that associated with C in the estimate returned by Algorithm 1. For the same reasons discussed above, only traces shorter than 30 steps were considered. The diagrams in Fig. $2(a, b)$ report this error under the open and closed world

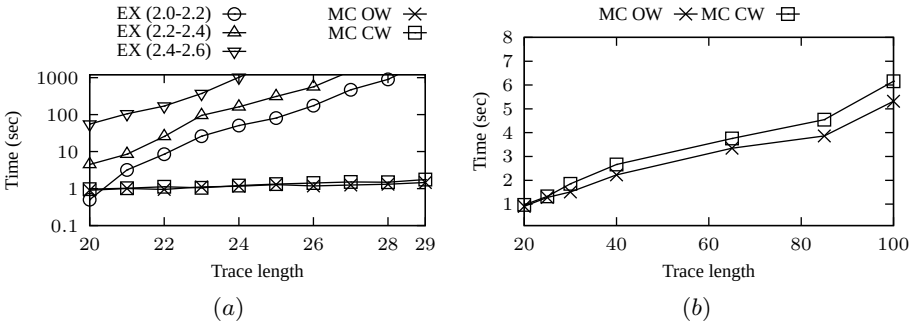


Fig. 1: (a): Execution times of EX, MC-OW and MC-CW vs. trace length over "short" traces; (b): Execution times of MC-OW and MC-CW vs. trace length over all the traces

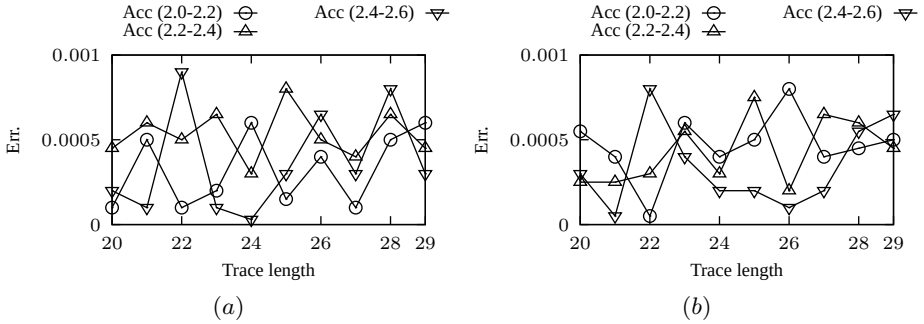


Fig. 2: Average accuracy of Algorithm 1 vs. trace length under the (a) open and the (b) closed world assumption (short traces only)

assumption, respectively. In each of these diagrams, three curves are reported, one for each set of traces $(X - Y)$ introduced above. These diagrams show that the average error is, for all the considered trace lengths, lower than the error threshold given as input to Algorithm 1 (we recall that we set $\epsilon = 0.001$, with confidence level $1-\alpha = 95\%$), and that the error is insensitive to both *ActPerEv*(Φ) and the trace length.

Discussion. The results show that our Monte-Carlo classification algorithm is generally faster than the exhaustive approach, and that, differently from the exhaustive approach, it is feasible even over "long" traces. Furthermore, the price to be paid is negligible, as the accuracy of our algorithm is very high. Note that we considered traces even longer than the standard business process traces: in fact, in the service agency dataset traces are rarely longer than 60 (this characteristics is shared with other real datasets, such as [11]). Interestingly, we point out that the low execution times of our algorithm make our approach useful both in offline and interactive analysis. The execution times of the order of seconds allow the possibility of using our algorithm for supporting the detection of security-breaches during the process monitoring in real time: in fact, the execution of any activity in a business process typically takes more than a few seconds.

6 Conclusions and future work

We have proposed a probabilistic approach exploiting the knowledge of process and security-breach models for classifying business log traces as process instances and/or potential security breaches. The framework can be straightforwardly extended to deal with different languages/mechanisms for defining the process and security-breach models: allowing more expressive composition rules or automata or Petri nets to specify these models simply requires to adapt function *checkModel*, that is orthogonal to the core of our technique. However, the impact of this modifications on the efficiency is worth investigating.

References

1. Van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE TKDE 16(9), 1128–1142 (2004)
2. van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science - R&D 23(2), 99–113 (2009)
3. van der Aalst, W.M., De Beer, H., van Dongen, B.F.: Process mining and verification of properties: An approach based on temporal logic. Springer (2005)
4. Accorsi, R., Stocker, T.: On the exploitation of process mining for security audits: the conformance checking case. In: Proc. of ACM SAC. pp. 1709–1716. ACM (2012)
5. Accorsi, R., Stocker, T., Müller, G.: On the exploitation of process mining for security audits: the process discovery case. In: Proc. of ACM SAC. pp. 1462–1468. ACM (2013)
6. Agresti, A., Coull, B.A.: Approximate is better than "exact" for interval estimation of binomial proportions. The American Statistician 52(2), 119–126 (1998)
7. Baier, T., Mendling, J., Weske, M.: Bridging abstraction layers in process mining. Information Systems 46, 123–139 (2014)
8. Baier, T., Rogge-Solti, A., Weske, M., Mendling, J.: Matching of events and activities - an approach based on constraint satisfaction. In: The Practice of Enterprise Modeling, Lecture Notes in Business Information Processing, vol. 197, pp. 58–72 (2014)
9. Bose, R., van der Aalst, W.M.: Discovering signature patterns from event logs. In: Symp. on Computational Intelligence and Data Mining (CIDM). pp. 111–118 (2013)
10. Cybenko, G., Berk, V.H.: Process query systems. IEEE Computer 40(1), 62–70 (2007)
11. van Dongen;, B.: Bpi challenge 2014: Activity log for incidents (2014), `http://dx.doi.org/10.4121/uuid:86977bac-f874-49cf-8337-80f26bf5d2ef`
12. Greco, G., Guzzo, A., Lupia, F., Pontieri, L.: Process discovery under precedence constraints. ACM Trans. Knowl. Discov. Data 9(4), 32:1–32:39 (2015)
13. Jans, M., van der Werf, J.M.E.M., Lybaert, N., Vanhoof, K.: A business process mining application for internal transaction fraud mitigation. Expert Syst. Appl. 38(10) (2011)
14. Lippmann, R.P., Ingols, K.W.: An annotated review of past papers on attack graphs. Tech. rep., DTIC Document (2005)
15. Rozinat, A., van der Aalst, W.M.: Conformance checking of processes based on monitoring real behavior. Information Systems 33(1), 64–95 (2008)
16. Sadiq, S.W., Orlowska, M.E., Sadiq, W.: Specification and validation of process constraints for flexible workflows. Inf. Syst. 30(5), 349–378 (2005)
17. Sauer, T., Minor, M., Bergmann, R.: Inverse workflows for supporting agile business process management. In: Wissensmanagement. pp. 204–213 (2011)
18. Suriadi, S., Weiß, B., Winkelmann, A., ter Hofstede, A.H., Adams, M., Conforti, R., Fidge, C., La Rosa, M., Ouyang, C., Rosemann, M., et al.: Current research in risk-aware business process management: overview, comparison, and gap analysis. CAIS 34(1), 933–984 (2014)

Redescription mining with multi-label Predictive Clustering Trees

Matej Mihelčić[1,3], Sašo Džeroski[2,3], Nada Lavrač[2,3], and Tomislav Šmuc[1]

[1] Ruđer Bošković Institute
Bijenička cesta 54, 10000 Zagreb, Croatia
{matej.mihelcic, tomislav.smuc}@irb.hr
[2] Jožef Stefan Institute
Jamova cesta 39, 1000 Ljubljana, Slovenia
{saso.dzeroski, nada.lavrac}@ijs.sl
[3] Jožef Stefan International Postgraduate School
Jamova cesta 39, 1000 Ljubljana, Slovenia

Abstract. Redescription mining is a field of knowledge discovery which aims to simultaneously find different descriptions of subsets of elements in the data. One of its useful properties is the ability to find connections between different sets of descriptive attributes and provide researchers with a more comprehensive set of rules allowing them to better understand the underlying problem. In this work, we present a novel algorithm for mining redescriptions based on multi-label Predictive Clustering Trees. This approach uses information about element membership in different generated rules to search for new redescriptions and is able to produce highly accurate, statistically significant redescriptions described by boolean, nominal or numeric attributes. As opposed to current tree-based approaches that use multi class classification, we use multi target classification or regression to create redescriptions. This allows us to create accurate cluster hierarchy from which we create multiple redescriptions that can contain overlapping elements. We discuss and illustrate the properties of the algorithm by extracting redescriptions from data describing 199 world countries based on their trading patterns and general information.

Keywords: knowledge discovery, redescription mining, predictive clustering trees, world trade

1 Introduction

Pattern mining [1, 11] aims at discovering relevant pieces of knowledge from a database. Researchers in this field have devoted a lot of effort to two very important problems: finding relevant patterns and generating relevant associations that will be presented to the user. Redescription mining shares these goals, but in addition tries to find different descriptions of the same patterns that will be presented to the user. It is an unsupervised, descriptive, knowledge discovery task with the goal of finding subsets of data that can be characterized with

multiple descriptions. The goal of such analysis is to find similarities between different elements and connections between different descriptive attribute sets (views) which ultimately lead to better understanding of the underlying data. The field of redescription mining was introduced by Ramakrishnan et al. [14]. In the same paper, they present a novel algorithm to mine redescriptions based on decision trees, called the CARTwheels. The algorithm works by building two decision trees (one for each view) that are joined in the leaves. Redescriptions are found by examining the paths from the root node of the first tree to the root node of the second and the algorithm uses multi class classification to guide the search between two views. Other approaches to mine redescriptions include the Zaki et al. [18] approach that uses a lattice of closed descriptor sets to find redescriptions. Further, Parida et al. [13] introduce algorithms for mining exact and approximate redescriptions, Gallo et al. [9] present the greedy and the MID algorithm based on frequent itemset mining, Galbrun et al. [6] present a novel greedy algorithm for mining redescriptions based on the greedy approach by Gallo et al. [9]. Finally, redescription mining is extended by Galbrun et al. to a relational [5] and an interactive setting [7].

Redescription mining is highly applicable in biology, economy, pharmacy, ecology and many other fields, where it is important to understand connections between different descriptors and to find regularities that are valid for different element subsets. Redescriptions are represented in the form of rules and the aim is to make these rules understandable and interpretable.

In this work, we present a novel algorithm for mining redescriptions based on multi-label predictive clustering trees [3, 12]. Our approach uses multi-label classification or regression to find highly accurate, statistically significant redescriptions, which differentiates it from other tree based approaches, especially the CARTwheels. With this approach, we are able to construct a cluster hierarchy from which it is possible to get different overlapping redescriptions. The approach is related to multi-view [2] and multilayer [10] clustering, though the main goal here is to find accurate redescriptions of interesting subsets of data, while clustering tends to find clusters that are not always easy to interpret. After introducing the necessary notation (Section 2), we present the algorithm, determine its computational complexity (Section 3) and use it to find descriptions of 199 different world countries based on their trading behaviour [16] and general country information [17] for the year 2012 (Section 4). The main focus is on rules containing only logical conjuntion operators, since these rules are the most interpretable and the knowledge represented is necessarily valid for all described elements. Finally, we conclude and outline directions for future work in Section 5.

2 Notation and definitions

Redescription mining in general considers redescriptions constructed on a set of views $\{W_1, W_2, \ldots, W_n\}$, however in this paper we use only two views $\{W_1, W_2\}$. The corresponding attribute (variable) sets are denoted by V_1 and V_2. Each

view contains $|E|$ rows and $|V_1|$, $|V_2|$ columns. Value $W_1(i,j)$ is the value of element e_i for the attribute a_j. The data $D = (V_1, V_2, E, W_1, W_2)$ is a quintuple of the attribute sets, element set, and the appropriate view mappings. A query (denoted q) is a logical formula F, where q_1 contains literals from V_1. The set of elements described by a query is called its support. A redescription $R = (q_1, q_2)$ is defined as a pair of queries, one for each view in the data. The support of a redescription is the set of elements supported by both queries that constitute this redescription $supp(R) = supp(q_1) \cap supp(q_2)$. We use $attr(R)$ to denote all the attributes contained in the redescription R. The accuracy of a redescription $R = (q_1, q_2)$ is measured using the Jaccard similarity coefficient.

$$JS(R) = \frac{|supp(q_1) \cap supp(q_2))|}{|supp(q_1) \cup supp(q_2)|}$$

The Jaccard similarity coefficient is not the only measure used in the field because it is possible to obtain redescriptions covering huge element subsets, that necessarily have very good overlap of their queries. In this cases it is preferred to have redescriptions that reveal some more specific knowledge about the studied problem that is harder to obtain by random sampling from the underlying data distribution. This is why we compute the statistical significance (p-value) of each obtained redescription. We denote the marginal probability of a query q_1, q_2 with $p_1 = \frac{supp(q_1)}{|E|}$, $p_2 = \frac{supp(q_2)}{|E|}$ respectively. We define the set of elements in the intersection of the queries with $o = supp(q_1) \cap supp(q_2)$. The corresponding p-value ([8]) is defined as

$$pV(q_1, q_2) = \sum_{n=|o|}^{|E|} \binom{|E|}{n} (p_1 \cdot p_2)^n \cdot (1 - p_1 \cdot p_2)^{|E|-n}$$

The p-value tells us if we can dismiss the null hypothesis that assumes that we obtained a given subset of elements by joining two random rules with marginal probabilities equal to the fraction of covered elements. If the obtained p-value is lower than some predefined threshold, called significance level, then this null hypothesis should be rejected. It is a somewhat optimistic criterion, since the assumption that all elements can be sampled with equal probability need not hold for all datasets. In addition, we perform multiple iterations of the algorithm and generate many rules at each iteration.

3 Constrained CLUS-RM algorithm

In this section, we describe a variant of the CLUS-RM algorithm for mining redescriptions that at each step improves redescription set of size constrained by the user. The algorithm uses multi-label predictive clustering trees (PCT) [3, 12] to create a cluster hierarchy that is later transformed into redescriptions. We start by providing and explaining the high level pseudo code of the algorithm (Algorithm 1) and then go into the details of each procedure in the algorithm.

Algorithm 1 The CLUS-RM algorithm

Input: First view data (W1), Second view data (W2), Settings file
Output: A set of redescriptions $\mathcal{R}$

```
 1: procedure CLUS-RM
 2:     DW1_init ← prepareTargetsForInitialPCT(W1)
 3:     DW2_init ← prepareTargetsForInitialPCT(W2)
 4:     PCTW1 ← createW1SideInitialPCT(DW1_init)
 5:     PCTW2 ← createW2SideInitialPCT(DW2_init)
 6:     RW1 ← extractRules(PCTW1)
 7:     RW2 ← extractRules(PCTW2)
 8:     initializeArrays(elFreq, attrFreq, redScoreEl, redScoreAt, numEx, numAttr,
                         numRetRed)
 9:     while RunInd<maxIter do
10:         TmpRW1 ← emptyRuleSet()
11:         TmpRW2 ← emptyRuleSet()
12:         D_W1Targ ← prepareTargets(RW2)
13:         D_W2Targ ← prepareTargets(RW1)
14:         PCTW1 ← createPCT(D_W1Targ)
15:         PCTW2 ← createPCT(D_W2Targ)
16:         TmpRW1 ← TmpRW1 ∪* extractRules(PCTW1)
17:         TmpRW2 ← TmpRW2 ∪* extractRules(PCTW2)
18:         RW1 ← RW1 ∪ TmpRW1
19:         RW2 ← RW2 ∪ TmpRW2
20:         R ← MineRed(RW1, RW2, expansionType,
                 ConstSet, iteration, opSet, elFreq, attrFreq, redScoreEl, redScoreAt)
21:     return R
```

The algorithm starts by creating initial clusters for both views (line 2 and 3 in Algorithm 1). Initial clusters are obtained by constructing one additional synthetic example for each example in the original view (see Figure 1). The artificial examples for the selected view are created from the original data by random shuffling attribute values between the examples. The shuffling step is repeated for each example $min(500, min(0.7 * |E|, |V_1| + |V_2|))$ times and exactly $max(1, 0.08 * (|V_1| + |V_2|))$ randomly selected attribute values from a randomly chosen element are copied to the constructed artificial example. The shuffling parameters where chosen so that we make enough changes to the artificial examples to brake the correlations between attribute values, but still do not introduce to many attribute values from each individual randomly chosen example. Original examples are assigned a target label of 1.0, while artificial examples are assigned a target label of 0.0. This procedure creates artificial examples for which it breaks the correlations among the attributes. Since we start from unlabelled data, we transform the originally unsupervised problem to a supervised one. The division between artificial and the original examples allows us to construct a cluster hierarchy, simultaneously creating descriptions of the original examples. The described procedure is one possible way to construct the initial clusters; other approaches include assigning a random target attribute or using clusters com-

puted by some other single or multi-view clustering algorithm. However, the initialization procedure used in our algorithm should preserve any strong (specific) connections and correlations that exist in the original data which might be broken by using an approach that assigns random target labels.

Entity	W1A1	W1A2	W1A3
E1	1.1	2.5	3.4
E2	1.5	2.2	4.0
E3	5.5	-0.6	-0.2
E4	4.4	-0.2	2.0
E5	3.2	1.7	2.9

(a) Original dataset for view 1

Entity	W2A1	W2A2	W2A3
E1	TRUE	FALSE	FALSE
E2	TRUE	TRUE	FALSE
E3	FALSE	FALSE	TRUE
E4	TRUE	TRUE	TRUE
E5	TRUE	FALSE	TRUE

(b) Original dataset for view 2

Entity	W1A1	W1A2	W1A3	Target
E1	1.1	2.5	3.4	1.0
E2	1.5	2.2	4.0	1.0
E3	5.5	-0.6	-0.2	1.0
E4	4.4	-0.2	2.0	1.0
E5	3.2	1.7	2.9	1.0
E1'	4.4	2.5	2.9	0.0
E2'	3.2	-0.6	4.0	0.0
E3'	3.2	-0.6	2.9	0.0
E4'	4.4	-0.2	4.0	0.0
E5'	5.5	1.7	2.9	0.0

(c) Initial dataset for view 1

Entity	W2A1	W2A2	W2A3	Target
E1	TRUE	FALSE	FALSE	1.0
E2	TRUE	TRUE	FALSE	1.0
E3	FALSE	FALSE	TRUE	1.0
E4	TRUE	TRUE	TRUE	1.0
E5	TRUE	FALSE	TRUE	1.0
E1'	TRUE	FALSE	TRUE	0.0
E2'	FALSE	FALSE	TRUE	0.0
E3'	TRUE	TRUE	TRUE	0.0
E4'	FALSE	TRUE	FALSE	0.0
E5'	FALSE	FALSE	TRUE	0.0

(d) Initial dataset for view 2

Fig. 1: Example tables describing the creation of artificial examples.

After creating the initial dataset, we build predictive clustering trees on both views by performing regression on the target label and using other attributes as descriptive. The use of regression trees is purely technical, since it generates more rules because of the additional threshold, associated with the target variable. These trees are converted to rules that describe element sets and are necessary for the next step of the algorithm. The rule lists RW1 and RW2 contain generated rules, and a new rule is added to the list if it differs from all other rules in a predefined number of attributes or if it describes a new unique element subset (the $\cup_*$ operator in Algorithm 1). The iterative process of the algorithm begins right after rule creation. Here, we create targets based on the rules obtained in the previous step or in the initialization step. Rules obtained by predictive clustering on W1 are used to build targets for clustering on W2 (denoted $W1T1$, $W1T2$), and vice versa. For each example in the dataset we assign label 1.0 if the example is described by some specific rule, otherwise 0.0 (see Figure 2). For example, the attribute $W2T1$ from dataset for view 1 represents the rule $IF\ W2A1 = TRUE$ (constructed on dataset for view 2), which describes elements $E1$, $E2$, $E4$, $E5$. By

placing this target attribute in the view 1 dataset, we guide the PCT construction to create a cluster containing and describing the same set of elements with view 1 descriptive variables (one choice that satisfies this condition is $IF\ W1A3 > 0$).

E	W1A1	W1A2	W1A3	W2T1	W2T2
E1	1.1	2.5	3.4	1.0	0.0
E2	1.5	2.2	4.0	1.0	0.0
E3	5.5	-0.6	-0.2	0.0	0.0
E4	4.4	-0.2	2.0	1.0	0.0
E5	3.2	1.7	2.9	1.0	1.0

(a) Dataset for view 1

E	W2A1	W2A2	W2A3	W1T1	W1T2
E1	TRUE	FALSE	FALSE	0.0	1.0
E2	TRUE	TRUE	FALSE	0.0	1.0
E3	FALSE	FALSE	TRUE	1.0	0.0
E4	TRUE	TRUE	TRUE	1.0	0.0
E5	TRUE	FALSE	TRUE	1.0	1.0

(b) Dataset for view 2

Fig. 2: Example tables describing target creation.

Rules obtained in the previous step are combined into redescriptions, which are considered, if they satisfy a given set of constraints $ConstSet$. The set of constraints consists of minimal Jaccard similarity index needed to add a redescription to the redescription set ($minJS$), maximum allowed p-value ($maxPval$) and minimum and maximum support ($minSupp,\ maxSupp$).

3.1 The procedure for creating redescriptions

Here, we introduce the algorithm for mining redescriptions (Algorithm 2). The procedure matches rules into redescriptions by joining a view 1 rules (or its negation, if allowed by the user) with a rules (or its negation) from view 2 (see Figure 3). We distinguish three cases of creating redescriptions from rules:

1. Unguided initial: $UInit \leftarrow (RW1 \times_{ConstSet}^{opSet\setminus\{\vee\}} RW2)$
2. Unguided: $U \leftarrow (RW1_{newRuleIt} \times_{ConstSet}^{opSet\setminus\{\vee\}} RW2_{newRuleIt})$
3. Guided: $G \leftarrow (RW1_{newRuleIt} \times_{ConstSet}^{opSet\setminus\{\vee\}} RW2_{oldRuleIt}) \cup$
   $(RW1_{oldRuleIt} \times_{ConstSet}^{opSet\setminus\{\vee\}} RW2_{newRuleIt})$

The $\times_{ConstSet}^{opSet}$ operator denotes a Cartesian product of two sets, allowing the use of logical operators from $opSet$ and leaving only those redescriptions that satisfy a given set of constraints $ConstSet$. The unguided expansion allows obtaining redescriptions with more diverse subsets of elements that can later be improved through the iteration process. The disjunction operator can be used to increase redescription accuracy and support by finding a complementing rule or its negation. For a redescription $R = (q_1, q_2)$, we find rules r that maximize:

1. $JS(supp(q_1 \vee r) \setminus supp(R), supp(q_2) \setminus supp(R))$
2. $JS(supp(q_1 \vee \neg r) \setminus supp(R), supp(q_2) \setminus supp(R))$
3. $JS(supp(q_1) \setminus supp(R), supp(q_2 \vee r) \setminus supp(R))$
4. $JS(supp(q_1) \setminus supp(R), supp(q_2 \vee \neg r) \setminus supp(R))$

Algorithm 2 MineRed

Input: Rule sets for view 1 and view 2, expansion type, ConstSet, iteration number, logical operator set, elFreq, attrFreq, redScoreEl, redScoreAt
Output: A set of redescriptions $\mathcal{R}$

procedure MineRed
  expansionSet ← ()
  if expansionType==unguidedExpansion AND iteration==0 **then**
    expansionSet ← UInit
  else if expansionType==unguidedExpansion AND iteration $\neq 0$ **then**
    expansionSet ← U
  else if expansionType==guidedExpansion **then**
    expansionSet ← G
  for $R' \in expansionSet$ **do**
    if $|\mathcal{R}|$<ConstSet.MaxRed **then**
      updateFrequencies(elFreq, attrFreq)
      $\mathcal{R} \leftarrow \mathcal{R} \cup R'$
      if $|\mathcal{R}|$ == ConstSet.MaxRed **then**
        for $R \in \mathcal{R}$ **do**
          computeScores(elFreq,attrFreq, redScoreEl, redScoreAt, R)
    else if $|\mathcal{R}|$ == ConstSet.MaxRed **then**
      compScore(elFreq,attrFreq, redScoreEl, redScoreAt, R')
      $R_b \leftarrow argmax_{R \in \mathcal{R} |\ R.pval \geq R'.pval}(\frac{(1.0-R'.elSc+1.0-R'.atrSc+R'.JS)}{3} - \frac{(1.0-R.elSc+1.0-R.attrSc+R.JS)}{3})$
      updateFrequencies(elFreq, attrFreq, R', R)
      updateScores(elFreq,attrFreq, redScoreEl, redScoreAt, R', R)
      $\mathcal{R} \leftarrow \mathcal{R} \backslash R_b \cup R'$
  for R$\in \mathcal{R}$ **do**
    if expansionType==unguidedExpansion AND iteration==0 **then**
      $r'_{W1} \leftarrow argmax(R.maxRef(r), R.maxRef(\neg r),\ r \in RW1)$
      $R_{ref} \leftarrow (r'_{W1} \vee R.rW1 \times R.rW2)$
      $r'_{W2} \leftarrow argmax(R_{ref}.maxRef(r), R_{ref}.maxRef(\neg r),\ r \in RW2)$
      $R_{ref} \leftarrow (R_{ref}.rW1 \times r'_{W2} \vee R.rW2)$
      updateFrequencies(elFreq, attrFreq, R, R_{ref})
      updateScores(elFreq,attrFreq, redScoreEl, redScoreAt, R, R_{ref})
      $\mathcal{R} \leftarrow \mathcal{R} \backslash R \cup R_{ref}$
    else
      $r'_{W1} \leftarrow argmax(R.maxRef(r), R.maxRef(\neg r),\ r \in RW1_{newRuleIt})$
      $R_{ref} \leftarrow (r'_{W1} \vee R.rW1 \times R.rW2)$
      $r'_{W2} \leftarrow argmax(R_{ref}.maxRef(r), R_{ref}.maxRef(\neg r), r \in RW2_{newRuleIt})$
      $R_{ref} \leftarrow (R_{ref}.rW1 \times r'_{W2} \vee R.rW2)$
      updateFrequencies(elFreq, attrFreq, R, R_{ref})
      updateScores(elFreq,attrFreq, redScoreEl, redScoreAt, R, R_{ref})
      $\mathcal{R} \leftarrow \mathcal{R} \backslash R \cup R_{ref}$
  return $\mathcal{R}$

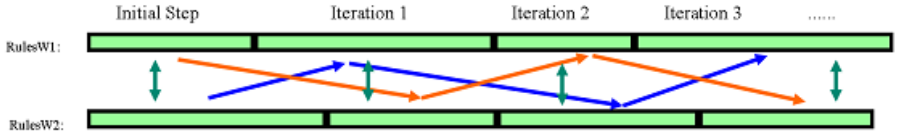


Fig. 3: Illustration of the iterative rule guidance through the exchange of targets between views (single-headed arrows). The two- headed arrows denote possible unguided rule matching.

Rule r is found so that it complements elements that are supported by q_2 but not by q_1 and vice versa. This process is denoted in the code with $maxRef(r, R)$. In effect, the algorithm finds first $numRed$ redescriptions and then iteratively enriches this set by exchanging the redescription with the worst comparative score with the newly created redescription. The algorithm uses 4 arrays (`elFreq`, `attrFreq`, `redScoreEl`, `redScoreAt`) to incrementally add and improve redescriptions in the redescription set. The element/attribute frequency arrays contain the number of times each element/attribute from the dataset occurs in redescriptions from a redescription set. Redescription scores are computed as $redScoreEl(R) = \sum_{e \in supp(R)} elFreq[e]$, and $redScoreAt(R) = \sum_{a \in atr(R)} attrFreq[a]$. The score of a new redescription is computed in the same way by using existing frequencies from the set. If the algorithm finds a redescription R' that is better than some redescription R_i from the redescription set, all arrays are updated in the following way: $\forall e \in supp(R),\ elFreq[e] - -$, $\forall a \in attr(R),\ attrFreq[a] - -$, $\forall e \in supp(R'),\ elFreq[e]{+}{+}$, $\forall a \in attr(R'),\ attrFreq[a]{+}{+}$. For each redescription $R'' \in \mathcal{R} \backslash R_i$, its score is recomputed as $\forall e \in supp(R_i),\ redScoreEl[R''] - -$, $\forall e \in supp(R'),\ redScoreEl[R'']{+}{+}$ and $\forall a \in attr(R_i),\ redScoreAt[R''] - -$, $\forall a \in attr(R'),\ redScoreAt[R''] - -$. This score favours redescriptions that describe elements with low frequency by using non frequent attributes. At the same time it tries to find accurate and significant redescriptions.

3.2 Algorithm time complexity

In this subsection we analyse the algorithm's time complexity. We start from the known results [15] that predictive clustering tree has the worst time complexity $O(z \cdot m \cdot |E|^2)$ to completely induce the tree, where m denotes the number of descriptive variables in a selected view and z the total number of internal nodes in the tree. The initialization step has complexity $O(|E| \cdot (|V_1| + |V_2|))$. Further, we use an algorithm of worst time complexity $O(z)$ to transform PCT to rules. Next, we compute the Cartesian product of two rule sets and extract those that satisfy the ConstraintSet defined by the user. The worst time complexity of creating a redescription is $O(n \cdot log(n'))$, where n equals the number of elements covered by the rule created on $W1$ and n' denotes the number of elements covered by the rule created on $W2$. In this step, we make $\sum_{i \in R_L} \sum_{j \in R_R} n_i \cdot log(n_j)$ element comparisons. Since it is true that $n \leq |E|$ and $n' \leq |E|$, the worst

time complexity of this step is $O(z^2 \cdot (|E| \cdot log(|E|)))$. However, this is a very conservative estimate. If we have a balanced tree, the complexity is closer to $O(z \cdot d \cdot |E| \cdot log(|E|))$, where d equals the tree depth. For each created redescription, we need to update the attribute, element frequency tables and if added the total redescription scores. In the worst case, the complexity of this step is $O(|\mathcal{R}| \cdot (|E| + |V_1| + |V_2|))$, however $|\mathcal{R}|$ is a predefined constant, since it is the number of redescriptions that should be returned to the user. We should also note here, that $O(|E| + |V_1| + |V_2|)$ is a very pessimistic boundary, since it claims that each redescription supports all elements in the dataset and contains all attributes (descriptive variables) from both views. In reality, it is not interesting to consider redescriptions that support all elements, this number is constrained by user defined variables $minSupport$, $maxSupport$, in addition it is crucial to have redescriptions that contain the smallest possible number of attributes. This number is less or equal to the tree depth for rules containing only conjunction operators. The computation of rules containing negation and disjunction operators has a complexity of $O(z \cdot |E| \cdot log(|E|))$. The total algorithm time complexity is: $O(|E \cdot (|V_1| + |V_2|)| + z \cdot |V_1| \cdot |E|^2 + z \cdot |V_2| \cdot |E|^2 + 2 \cdot z + z^2 \cdot |E| \cdot log(|E|) \cdot (|E| + |V_1| + |V_2|) + 2 \cdot z \cdot |E| \cdot log(|E|))$ which is in fact $O(z \cdot (|V_1| + |V_2|) \cdot |E|^2 + z^2 \cdot |E| \cdot log(|E|) \cdot (|E| + |V_1| + |V_2|))$. The procedure is repeated $numIter$ times, but this number is a user defined constant that does not change the overall complexity. There are a number of optimizations that can be employed to reduce the complexity of computing redescriptions. One is to use rule indexing to combine only those rules that are certain to cross the user defined threshold, the other one is to use Local Sensitive Hashing [4]. There are advantages of using these framework, especially in the cases where we allow for soft constraints that can be changed adaptively depending on the underlying data.

4 Mining redescriptions on data describing countries

We present the experimental results of mining redescriptions with our algorithm on data describing 199 world countries. The dataset contains two views, both containing numerical attributes with possible missing values. One view contains 312 attributes representing the percentage of import and export of commodities for countries in the year 2012, while the second view contains 51 attribute containing general country information obtained from the World Bank for the same year. The algorithm was tested with various number of iterations (100, 200, 300). For each fixed number of iterations, we performed 10 runs of the algorithm, computed redescription sets containing 50 redescriptions and measured the average Jaccard similarity index and average redescription supports. Allowed redescription supports were in the range $[5, 120]$, the maximum allowed p-value was equal to 0.01 and the minimum Jaccard similarity index was equal to 0.6. As we can see from Figure 4, with an increased number of iterations, the algorithm finds redescriptions with higher accuracy, but describing smaller subsets of countries. The mean value of the total overall coverage of elements in the redescription set

varies between 45% and 52%. This indicates that the algorithm managed to find highly accurate redescriptions describing a significant number of total elements from the dataset.

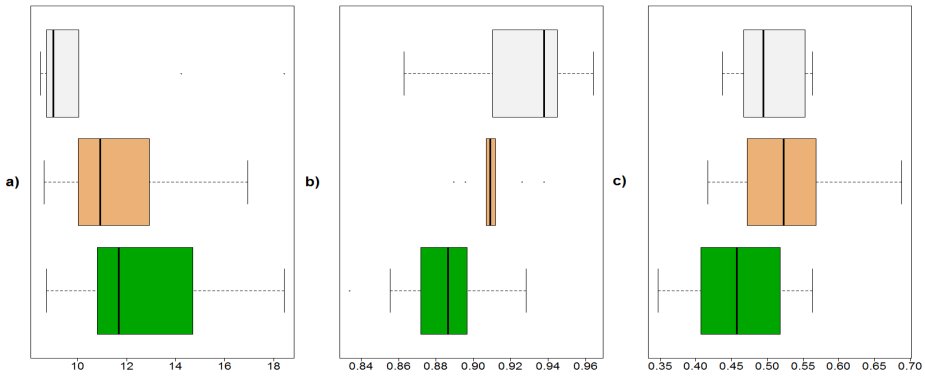


Fig. 4: A summary of the results for different numbers of algorithm runs (top to bottom: 300, 200, 100): average redescription support size a), average Jaccard similarity index b), fraction of all elements described by a redescription c).

Next, we analyse one of the obtained redescription and present several other examples (Table 1). This table demonstrates the complex relations between redescription support size, rule complexity, JS and p-value.

```
W1R: MON_GROWTH >= -4.7851 <= 12.791 AND POP_64 >= 14.0491 <= 20.8229
     AND UNEM_M >= 4.3 <= 15.0
W2R: I_MEAT_PROD >= 1.0 <= 2.0 AND E/I_IND_MACH >= 0.654 <= 3.146 AND
     E/I_MAN_GOODS >= 0.739 <= 1.455 AND E/I_CER >= 0.604 <= 15.845
     AND E_HSTI_MAN >= 8.0 <= 29.0
```

This redescription describes 19 world countries (Austria, Belgium, Czech Republic, Denmark, Finland, France, Germany, Italy, Netherlands, Slovenia, Spain, Sweden, Switzerland and United Kingdom) with Jaccard similarity 1.0. We found that the annual money and quasi money growth of these countries ranges from $[-4.8, 12.8]$%, the percentage of population with age 65 or above is in $[14.0, 20.8]$%, and the male unemployment rate is in $[4.3, 15]$%. At the same time, these countries share the following trade properties: the import of meat and meat preparations constitutes $[1, 2]$% of the total country import, the ratio of export and import of other industrial machinery and parts is in the interval $[0.7, 3.1]$, this ratio for manufactured goods is in $[0.7, 1.5]$, the ratio for cereals and cereal preparations is in $[0.6, 15.8]$ and the export of high skill and technology intensive manufactures forms $[8, 29]$% of the total export of these countries. This is a statistically highly significant redescription with a p-value of $4.8 \cdot 10^{-15}$, it contains 3 descriptive variables for view 1 and 5 variables for view 2. It is a medium size redescription, based on its rule size.

Table 1: Redescription examples of different support sizes.

POP_GROWTH >= 0.0939 <= 1.0655 AND POP_64 >= 16.2121 <= 21.1009 I_CER >= 0.0 <= 2.0 AND E/I_SPEC_MACH >= 1.051 <= 4.308 AND I_CHEM_PROD >= 11.0 <= 23.0	**Jaccard coefficient**: 1.0 **Redescription support**: 14 **Redescription p-value**: $1.5 \cdot 10^{-13}$
UNEMPL_LONG >= 0.0 <= 9.8 I_LIVE_AN >= 0.0 <= 0.0 AND E/I_IND_MACH >= 0.394 <= 3.146 AND I_APCL_ACC >= 1.0 <= 7.0	**Jaccard coefficient**: 0.75 **Redescription support**: 40 **Redescription p-value**: $1.9 \cdot 10^{-13}$
MON_GROWTH >= -0.3463 <= 57.8338 AND POP_64 >= 0.3573 <= 8.7588 AND LABOR_PARTICIP_RATE >= 43.7 <= 86.7 AND CRED_COVER >= 0.0 <= 46.0 AND M2 >= 11.9206 <= 113.8993 E/I_MEDPH_PROD >= 0.0 <= 0.248 AND E/I_METW_MACH >= 0.0 <= 0.151 AND E_NMMIN_MAN >= 0.0 <= 34.0 AND E/I_FOOD_BASIC >= 0.001 <= 2.429 AND E_ALLOC_PROD >= 94.0 <= 100.0 AND E/I_LSTI_MAN >= 0.009 <= 3.469	**Jaccard coefficient**: 0.82 **Redescription support**: 68 **Redescription p-value**: $7.0 \cdot 10^{-13}$

5 Conclusion

We have presented a novel algorithm for redescription mining, based on multi-label predictive clustering trees. This approach uses information about element membership in generated rules to construct redescriptions that are further used to incrementally improve the redescription set of a user defined size. We analysed the algorithm time complexity and evaluated its performance on data describing world countries. The results show that there are benefits of using more iterations. The generated redescriptions were statistically relevant with very low p-values (less than 10^{-5}). Many generated rules contained the maximum number of 6 attributes per rule in a redescription. In future work, we plan to extend the framework by deploying RF of PCTs which should allow the creation of a much larger number of different, diverse and at the end higher quality redescriptions. Further, we will focus on obtaining more accurate conjunctive rules, evaluate the algorithm on several datasets and compare it with other algorithms in the field. Finally, we intend to perform a more comprehensive evaluation of redescription sets, which should help users in evaluating results of various redescription mining algorithms.

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Databases, In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 207–216, Washington, D.C. (1993)
2. Bickel, S., Scheffer, T.: Multi-View Clustering. In Proceedings of the Fourth IEEE International Conference on Data Mining, pp. 19–26, Washington. (2004)
3. Blockeel., H.: Top-down Induction of First Order Logical Decision Trees. Phd thesis, Universiteit Leuven, Department of Computer Science. (1998)
4. Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J., D., Yang, C.: Finding interesting associations without support pruning, In ICDE, 489–499. (2000)
5. Galbrun, E., Kimmig, A.: Finding relational redescriptions. Machine Learning, 225–248 (2014)
6. Galbrun, E., Miettinen, P.: From black and white to full color: extending redescription mining outside the Boolean world. Statistical Analysis and Data Mining, 284–303 (2012)
7. Galbrun, E., Miettinen, P.: A Case of Visual and Interactive Data Analysis: Geospatial Redescription Mining. Instant Interactive Data Mining Workshop @ ECML-PKDD (2012)
8. Galbrun, E.: Methods for Redescription mining, University of Helsinki. (2013)
9. Gallo, A., Miettinen, P., Mannila, H.: Finding Subgroups having Several Descriptions: Algorithms for Redescription Mining. In Proceedings of the SIAM International Conference on Data Mining, pp. 334-345, Atlanta, Georgia (2008)
10. Gamberger, D., Mihelčić, M., Lavrač, N., Multilayer Clustering: A Discovery Experiment on Country Level Trading Data. In Proceedings of the 17th International Conference on Discovery Science, Lecture Notes in Computer Science, pp. 87–98, Bled. (2014)
11. Giacometti, A., Li, D. H., Marcel, P., Soulet, A.: 20 Years of Pattern Mining: A Bibliometric Survey, SIGKDD Explor. Newsl., 41–50 (2014)
12. Kocev, D., K., Vens, C., Struyf, J., Džeroski, S.: Tree ensembles for predicting structured outputs. Pattern Recognition, 817–833 (2013)
13. Parida, L., Ramakrishnan, N.: Redescription Mining: Structure Theory and Algorithms. In Proceedings of the 20th National Conference on Artificial Intelligence, pp. 837-844, Pittsburgh, Pennsylvania (2004)
14. Ramakrishnan, N., Kumar, D., Mishra, B., Potts, M., Helm, R. F.: Turning CARTwheels: an alternating algorithm for mining redescriptions. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp. 266-275, Seattle, WA (2004)
15. Stojanova, D., Ceci, M., Appice, A., Džeroski, S.: Network regression with predictive clustering trees, Data Mining and Knowledge Discovery, 378–413. (2012)
16. UNCTAD database, `http://unctadstat.unctad.org/EN/`.
17. World Bank database, `http://data.worldbank.org/`.
18. Zaki, M. J., and Ramakrishnan, N. Reasoning about sets using redescription mining. In Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp. 364–373, Chicago, Illinois (2005)

Generalizing Patterns for Cross-domain Analogy

F. Leuzzi[1] and S. Ferilli[1,2]

[1] Dipartimento di Informatica – Università di Bari
{fabio.leuzzi, stefano.ferilli}@uniba.it
[2] Centro Interdipartimentale per la Logica e sue Applicazioni – Università di Bari

Abstract. Analogy is the cognitive process of matching the characterizing features of two different items. This may enable reuse of knowledge across domains, which can be helpful to solve problems. Analogy is strongly related to semantics, because the mappings are based on the role and meaning of the features, which goes beyond simple syntactic association. The analogical mappings found between pairs of descriptions can be used to obtain more general analogical patterns. Such patterns may be stored in the long term memory, allowing self-improvement and growth. This paper proposes generalizations of patterns obtained by analogy, carried out through two main steps: (1) isolating analogous roles of two descriptions coming from different domains, and (2) abstracting from portions of knowledge that have no analogical relationships. The result is a multi-strategy approach in which the analogy brings to a generalization, that is, in turn, a novel description to reason over and over again. An example is provided to show the behavior and effect of the proposed generalization approach.

1 Introduction

Analogy is the cognitive process of matching the characterizing features of two different items (subjects, objects, situations, etc.). While it is often confused with similarity, there is a significant difference between them. Similarity maps exactly the same features in the two items. In analogy, a feature in one item can be mapped onto a completely different feature in the other, provided that they both play in some sense 'the same role' in the respective items. So, analogy is much related to *abstraction*, while similarity is more related to generalization. Indeed, abstracting the 'role' of the features away from their specific embodiment in the single items is fundamental to recognize the possibility of an analogical mapping between them. In some sense, similarity is a (simpler) kind of analogy. It is clear that analogy has a tight relationship to semantics, because the mappings are based on the role and meaning of the features in the two items, which goes beyond simple syntactic association (as for similarity). It allows one to reuse knowledge from a known domain to an unknown one, without having to learn from scratch. In fact, after finding the analogy on some (fundamental) features, the association can be extended to further features, for which experience has not yet discovered the association of roles.

In everyday life people often face problems about which they have no experience. Sometimes they call a friend for help, or call an expert that is able to solve the problem. However, sometimes they are able to identify an analogous experience (or any other type of reliable knowledge), from which inferring an hypothetical solution. The inferred solution is not always applicable (e.g., it might seem unreasonable), nevertheless it may provide an unexpected escape or the opportunity to learn something. Anyway, reasoning by analogy is essential for producing new conclusions that are helpful to solve a problem [3]. Several perspectives make this type of inference a primary issue: (1) in the study of learning, analogies are important in the transfer of knowledge and inferences across different concepts, situations, or domains; (2) analogies are often used in problem solving and reasoning; (3) analogies can serve as mental models to understand new domains; (4) analogy is important in creativity (e.g., it was a frequent mode of thought for such great scientists as Faraday, Maxwell, and Kepler); (5) analogy is used in communication and persuasion; (6) analogy and its cousin, similarity, underlie many other cognitive processes. [3] defined analogy as a partial similarity between different situations that supports further inferences. More precisely, it is a kind of similarity in which the same system of relations holds across different objects. Thus, analogies capture parallelisms across different descriptions (typically, one referring to a past experience and one referring to the situation under consideration). For short, we will call the description coming from prior knowledge, or the domain it is referred to, the *base*, and the description of the current problem, or the domain is referred to, as the *target*.

Typical analogies happen between pairs of specific situations. While this is already a very powerful way of immediately reusing knowledge to solve a problem in a new domain, it may happen that some abstract patterns can be reused across many domains. Since these represent solution schemes whose usefulness has been thoroughly tested, they may be quite promising also in future situations. Albeit not very frequent, these patterns are really valuable and it would be very important to recognize and store them when one comes across them. In practice, this means finding some kind of relevant generalization among different specific analogies. Of course, if one analogy is already a quite complex inference to make, generalization among analogies is much more difficult, because several similar parallelisms must be detected.

This paper proposes the generalization of patterns obtained by analogy, carried out through two main steps: (1) isolating analogous roles of two descriptions coming from different domains, and (2) abstracting from portions of knowledge that have no analogical relationships. The result is a meta-pattern for analogy that generalizes experiences coming from different domains. We propose a multi-strategy approach in which the analogy brings to a generalization, that is, in turn, a novel description to reason over and over again.

The remainder of this paper is organized as follows. In Section 2 related works are reviewed and criticized, and preliminary information is provided. Section 3 recalls our procedure to make analogy and subsequent inference. The general-

ization approach, along with a sample case, is provided in Section 4. Finally, Section 5 concludes the paper.

2 Preliminaries

According to [14], analogical reasoning involves an inductive step, that hypothesizes the presence of an analogy between two contexts, and a deductive step, that performs truth-preserving reasoning based on the inductively inferred knowledge. However, induction, defined as the process of inferring general knowledge from specific observations, does not fit reasoning by analogy. It would be better described as an *intuition*, since it generates a set of possible mappings that provide analogies with respect to one or more points of view.

Given a point of view, a set of analogical mappings can be used to identify a *recurring meta-pattern* (i.e. a common network of roles). When accomplishing a task, this allows to use the knowledge present in a more familiar base domain to enrich the knowledge of the target domain. The recurring meta-pattern can be seen as an abstract description of the schema shared by the domains under consideration. Unfortunately, the identification of an abstract theory describing several domains is quite complex, since abstractions based on syntactic transformations only might generate an inconsistent set of abstract clauses, even if the ground set is consistent [5].

Most research on analogy operators used formal (Propositional or First-Order) Logic as the most suitable representation for describing high-level cognitive and reasoning processes.

The *Structure Mapping Engine* (SME) [2] uses a local-to-global strategy to structurally align the base and the target, guided by a set of 'programmable' rules. This ensures great flexibility (e.g., allowing to encode similarity or metaphors [2, 4]), but requires additional knowledge (e.g., about commutativity). Its formalism [4] involves typed entities that must be declared, and can be translated into ground Horn clauses. It can process 'second order' relations (i.e., relations among relations).

ACME [9] implements a 'cooperative' procedure for parallel satisfaction of a set of interacting constraints[1] represented as a network of supporting and competing hypotheses about what elements to map. The constraints may be of three types: *structural*, satisfied when an exact isomorphism is detected between the analogues; *semantic similarity* supports possible correspondences between elements to the degree that they have similar meaning; *pragmatic centrality* favors correspondences that are pragmatically important to the analogist, either because a particular correspondence between two elements is presumed to hold, or because an element is judged to be sufficiently central that some mapping for it should be found.

LISA [8] builds symbolic representations in neurally inspired computing architectures, an approach named 'symbolic connectionism' [10, 7]. Such represen-

[1] This connectionist approach to constraint satisfaction was investigated in [15].

tations are claimed to give LISA the ability to bind roles to their fillers dynamically (i.e., at need), and to represent the resulting bindings independently of the roles and fillers themselves. LISA provides a natural account of the neural processes, also regarding *working memory* and *long term memory*. The mappings in LISA are sequential, so each one will influence the next; this helps to keep the soundness of the analogies.

DORA [1] performs four basic operations: retrieval of propositions from long-term memory, analogical mapping of propositions currently in working memory, intersection discovery for predication and refinement, and linking of role-filler sets into higher arity structures via self-supervised learning. Just like other works, it assumes that objects and relational roles have a shared pool of basic representational features.

STAR-2 [17] finds analogies by sequentially focusing on parts of the domain. The best mapping for the arguments of the propositions is obtained using parallel computation in a constraint satisfaction network, in order to cope with the explosion of the number of units needed for tensor product representation.

BART [13] performs learning and inference through a Bayesian model. It takes as input vectors representing objects, so that all of a model's relational knowledge must be acquired from non-relational inputs. In this work, the authors focus on learning from positive example only, justifying such a choice as a good approximation of learning in children. Furthermore, since children's learning of relations is often guided by linguistic input from adults, BART focuses on supervised learning using labeled examples.

Finally, Copycat [6] discovers analogies trying to operate in a psychologically realistic way, in a more general framework that simulates fluid concepts and cognitive fluidity. It works on character strings of the form $EFG : MNO = EFH : ?$. The main idea is that high-level cognition-like features emerge from the independent activity of many parallel processes. It is composed by: the Snippet, a kind of archive containing the types of concepts with which the system can work; the Workspace, in which several components cooperate like in a multi-agent system; the Coderack, that provides other agents waiting to be invoked stochastically to carry out sub-tasks into the Workspace, simulating fluidity and creativity.

Only a few works (e.g. DORA) proposed a learning strategy. This is a fundamental issue, since the analogy power resides in the special way in which the cross-domain generalization carries out the inductive step, that represents the quick fix to the lack of experience in the target domain that allows to learn.

3 Analogy and Inference

Our analogical engine can work both in a free setting, trying to find any kind of parallelism between two descriptions, or in a focused setting, where the experimenter can express an initial association between objects in the descriptions and the system must find relevant parallelism involving that association. We refer the interested reader to [11] for details about the underlying algorithms. Here, since

we are interested in developing a subsequent phase of processing, we will just briefly recall its main features and general behavior using a running example.

3.1 Representation Formalism

In our approach, each piece of knowledge is formally represented as a Horn clause [12], i.e. a disjunction of literals involving at most one positive literal, where a literal is a (possibly negated) atom. An atom is a predicate applied to its arguments, that are terms (in our case, only constants). A predicate p requiring n arguments is denoted as p/n. Implicitly assuming the inclusive disjunction operator, a clause can also be seen as a set of literals $\{l_0, \neg l_1, \ldots, \neg l_n\}$. The ProLog representation of a clause is l_0 :– $l_1, \ldots, l_n$; where, in the usual interpretation, l_0 is the *head* (i.e., the conclusion of an implication) and $l_1, \ldots, l_n$ is the *body* (i.e., the conjunction of premises of the implication). We may extract the predicate on which an atom $l = p(t_1, \ldots, t_n)$ is built using function $\text{predicate}(l) = p/n$, and the terms of l using function $\text{terms}(l) = \{t_1, \ldots, t_n\}$. These functions may be straightforwardly extended to literals, while for a clause C we define $\text{predicates}(C) = \cup_{l \in C}\{\text{predicate}(l)\}$ and $\text{terms}(C) = \cup_{l \in C}\text{terms}(l)$.

For our purposes, clauses are not interpreted in the usual way, but they just provide a suitable formalism for expressing the data. The l_i's express properties of, or relationships among, (the objects denoted by) their arguments. Analogical mappings are to be found in the body; the predicate in the head labels the situation that is being described in the body. The heads may be used to provide a preferred focus, i.e. a specific perspective for which the analogical mapping is sought. If exploited, this feature allows us to reduce the search space and direct the operations toward a particular goal of interest. The 'preferred perspective' is enabled when the arity of the predicates in the heads is the same, in which case the system is bound to establish an analogy between corresponding arguments. Using 0-ary predicates in the heads disables this feature. Let us show the formalism with a running example that will be used throughout the paper to illustrate the various steps of the procedure.

Example 1 (Fairy tale and life context). Let us consider the fairy tale *The fox and the grapes* (Walter Crane's version, in *Baby's Own Aesop*, 1887) "This fox has a longing for grapes: he jumps, but the bunch still escapes. So he goes away sour; and, 'tis said, to this hour declares that he's no taste for grapes." would be formalized as:

fairy_tale(fox, grape) :– cannot_reach(fox, grape, fox_does_not_reach_grape),
wants(fox, grape), is(fox, sour), has(fox, bad_opinion),
cause(fox_does_not_reach_grape, bad_opinion),
says(fox, grape_is_no_taste), says(fox, she_is_smart),
is(grape, not_ripe, grape_is_no_taste).

It means that people tends to belittle things that they would like to obtain but that they cannot, e.g., as in a situation where "John loves Carla but cannot have her, so he spreads a bad opinion about her":

situation(john, carla) :– cannot_have(john, carla, john_cannot_have_carla),
loves(john, carla), says(john, carla_is_bad),
says(john, he_is_a_charming_man),
is(carla, bad, carla_is_bad), feels(john, sad).

The heads in these descriptions suggest that we want to establish an analogy between the *fairy tale* and the given *situation*, in which John plays the role of the fox, and Carla plays the role of the grape. Note that literals *says(fox, she_is_smart)* and *says(john, he_is_a_charming_man)* are useless for the analogy.

Note also that this case needs to represent that the fox has a bad opinion about the grapes because he cannot reach them. But such causal relation, in turn, involves the relation between the fox and the grape (because the former cannot reach the latter), relating it to the bad opinion that the fox has got. Such a situation requires the use of a third argument representing the name of the whole concept for which the fox cannot reach the grape.

The two domains may cross-fertilize each other, providing each other pieces of knowledge that allow to better understand the described situation and help to accomplish tasks (e.g., making comparisons, solving problems, etc.) in them. To build an analogy, an analysis of the relationships in which the objects in the description are involved is fundamental.

3.2 Analogy

In a nutshell, our analogical reasoner initializes the analogy mapping and then progressively expands it, guided by linkedness (i.e., term sharing) among literals. New term or predicate associations are added, as long as they ensure overall consistency of the mapping. The outcome of an analogy is a pair of mappings, one concerning predicates (θ_P) and the other concerning terms (θ_T). Initially, the global predicate mapping is empty ($\theta_P = \emptyset$). Then, if a preferred perspective is expressed by the heads, the mapping of the heads' arguments is taken as a starting point. Suppose that the two clauses C' and C'' input to the procedure have heads l'_0 and l''_0, respectively. If a consistent one-to-one mapping between the arguments of l'_0 and l''_0 exists, then the global term mapping θ_T is initialized to such a mapping. Otherwise, an empty global term mapping $\theta_T = \emptyset$ is initially set, and candidate starting points must be identified by evaluating the shared knowledge between the two descriptions, that provides potential points of contact between them.

Following with the *Fairy tale and life context*, since the heads have the same arity, their terms are mapped, yielding the starting point

$$\theta_T = \{(\text{fox,john}), (\text{grape,carla})\} \qquad \theta_P = \emptyset$$

Then, the expansion phase starts, resulting in the final overall mapping reported in Table 1.

Table 1. Analogy between fairy tale and life context.

Mapped Predicates		**Mapped Terms**	
Base clause (fairy tale)	Target clause (life context)	Base clause (fairy tale)	Target clause (life context)
says/2	says/2	fox	john
is/3	is/3	grape	carla
wants/2	loves/2	grape_is_no_taste	carla_is_bad
cannot_reach/3	cannot_have/3	not_ripe	bad
is/2	uses/2	fox_does_not_reach_grape	john_cannot_have_carla
		sour	sad
		she_is_smart	he_is_a_charming_man

3.3 Inference and Re-representation

One-to-one alignment of analogous roles for entities and relationships across domains is of primary importance, because it ensures that part of the structural consistency is verified [2]. For this reason, it is taken as the basis for the inference step, aimed at transferring missing knowledge across the domains. The literals that are completely mapped (i.e., having their predicate and all of their arguments mapped by the analogy), but whose counterpart is not present in the other description, can be immediately projected onto the other domain by taking their analogous counterparts. Moreover, the inference can be extended to partially mapped literals, introducing new names for the missing elements (that represent new knowledge that can be hypothesized in the other domain). We identify these names with the '*skolem_*' prefix. Of course, the more Skolem elements in a projection, the less reliable that projection.

In the *Fairy tale and life context* running example, the expected explanation of the phenomenon is: "John has a bad opinion about Carla because he cannot have her love". The inference hypotheses from the fairy tale to the life context are:

1: *skolem_cause(john_cannot_have_carla, skolem_bad_opinion)*
2: *skolem_has(john, skolem_bad_opinion)*

that fully satisfy the expected interpretation.

4 Analogical Pattern Generalization

The analogical mappings found between pairs of descriptions representing experiences, contexts or concepts can be used to obtain more general analogical schemes. Specifically, each analogy can be 'condensed' in a pattern, that in turn can be used for searching further analogies with other experiences. Such a pattern is stored in the long term memory, allowing self-improvement and growth.

Table 2. Literal mappings between fairy tale and life context.

Fairy tale	Life context
fairy_tale(fox, grape)	*situation(john, carla)*
says(fox, grape_is_no_taste)	says(john, carla_is_bad)
says(fox, she_is_smart)	says(john, he_is_a_charming_man)
is(grape, not_ripe, grape_is_no_taste)	is(carla, bad, carla_is_bad)
wants(fox, grape)	loves(john, carla)
cannot_reach(fox, grape, fox_does_not_reach_grape)	cannot_have(john, carla, john_cannot_have_carla)
is(fox,sour)	feels(john,sad)

4.1 Formal Definition

Formally, let us consider two atoms having the same arity $n > 0$, $l' = p'(t'_1, \ldots, t'_n)$ and $l'' = p''(t''_1, \ldots, t''_n)$. We define their *atomic analogy* as the pair $\mathrm{a}(l', l'') = \langle \mathrm{a_P}(l', l''), \mathrm{a_T}(l', l'') \rangle$ where: (1) $\mathrm{a_P}(l', l'') = \{(p'/n, p''/n)\}$ is the *predicate analogy* between l' and l'', (2) $\mathrm{a_T}(l', l'') = \{(t'_1, t''_1), \ldots, (t'_n, t''_n)\}$ is the *term analogy* between l' and l''; if $\mathrm{a_T}(l', l'')$ is a one-to-one term mapping. In all other cases it is undefined.

Then, given an analogy $\Theta = \langle \theta_T, \theta_P \rangle$ between two clauses C' and C'', a generalized pattern C with $|C| \leq \min(|C'|, |C''|)$ is outlined as follows. $\forall (l', l'') \in C' \times C''$, $l' = p'(t'_1, ..., t'_n)$, $l'' = p''(t''_1, ..., t''_n)$, for which $\exists \mathrm{a}(l', l'')$ s.t. $\mathrm{a_P}(l', l'') \subseteq \theta_P \wedge \mathrm{a_T}(l', l'') \subseteq \theta_T : l = p(t_1, ..., t_n) \in C$, where: if $p' = p''$, then $p = p' = p''$, otherwise p is a new predicate ($p \notin \text{predicates}(C') \cup \text{predicates}(C'')$); $\forall i = 1, \ldots, n$: if $t'_i = t''_i$, then $t_i = t'_i = t''_i$, otherwise t_i is a new term ($t_i \notin \text{terms}(C') \cup \text{terms}(C'')$).

In our running example, the mapping in Table 1 yields the alignment of literals shown in Table 2, from which the following pattern is obtained:

pattern(fairy_tale(fox, grape), situation(john, carla)) :–
wants_OR_loves(fox_OR_john, grape_OR_carla),
cannot_reach_OR_cannot_have(fox_OR_john, grape_OR_carla,
fox_does_not_reach_grape_OR_john_cannot_have_carla),
says(fox_OR_john, grape_is_no_taste_OR_carla_is_bad),
says(fox_OR_john, she_is_smart_OR_he_is_a_charming_man),
is(grape_OR_carla, not_ripe_OR_bad, grape_is_no_taste_OR_carla_is_bad).

where the new predicates and terms have been named by chaining the names of the predicates and terms they generalize, just to let the reader trace back their meaning and role in the original analogy.

As in usual generalization, patterns can be used to find analogies with other descriptions (or even with other patterns). If such analogies do not fully map the pattern components, new (and more general) patterns (actually, *meta-patterns*) can be generated. Differently from usual generalization, new patterns do not replace the patterns from which they originated, because each analogy is motivated by a specific perspective, and so the corresponding pattern must be preserved as a representative of that perspective. So, the same (meta-)pattern may give rise to

several meta-patterns, based on different combinations of specific domains. This reflects the fact that different aspects of a given situation may have different analogies with other experience according to different perspectives. Note that, as long as (meta-)patterns are progressively generalized (or used with full analogies) with new domains, the surviving elements in the resulting meta-pattern are more and more supported and confirmed, and so they are likely to represent common sense knowledge. In particular, if specific predicate or term names survive many refinements, they are likely to represent fundamental concepts.

The history of a pattern can be traced by recording the origin of each predicate/term in the pattern using 4-tuples of the form:

(head, type, pattern_name, original_name)

where *head* stands for the head of the original clause, *type* indicates if the record concerns a predicate or a term, *pattern_name* represents the name reported in the pattern and *original_name* reports the name in the original clause.

4.2 Evaluation

Traditionally, the evaluation of analogy-related algorithms has been qualitative rather than quantitative. This is mainly because the most interesting thing is whether and how a proposed algorithm can catch relevant parallelisms between descriptions and propose interesting associations. Counting how many successful analogies an algorithm can return can be also quite tricky, since different analogies may be considered as successful depending on the perspective, and thus there is no definite notion of accuracy. As a consequence, no benchmark datasets have been developed on which running experiments. The literature has focused on showing the performance of algorithms on specific relevant cases. We will follow this stream of evaluation.

Solar System vs. Rutherford's Atom The analogy between *Solar system* and *Rutherford atom* has been widely used in the literature (e.g., in [2, 16]). Expressed in our formalism, we have the following descriptions:

solar_system(sun, planet) :– inanimate(sun), inanimate(planet), mass(sun, mass_sun), mass(planet, mass_planet), greater(mass_sun, mass_planet, major_mass), attraction(sun, planet, attracts), revolve_around(planet, sun, revolve), attraction_mass(major_mass, attracts, major_mass_ attracts), cause(major_mass_attracts, revolve, cause_revolve), temperature(sun, temp_sun), temperature(planet, temp_planet), greater(temp_sun, temp_planet, major_temp), gravity(mass_sun, mass_planet, force_gravity), cause(force_ gravity, attracts, why_attracts).

rutherford_atom(nucleus, electron) :– inanimate(nucleus), inanimate(electron), mass(nucleus, mass_n), mass(electron, mass_e), greater(mass_n, mass_e, major_mass), attraction(nucleus, electron, attracts), revolve_around(electron, nucleus, revolve), charge(electron, q_electron), charge(nucleus, q_nucleus), opposite_sign(q_nucleus, q_electron, major_charge), cause(major_charge, attracts, why_attracts).

Table 3. Solar system and Rutherford atom mapping provided by RAM.

Mapped predicates		**Mapped terms**	
Base clause	Target clause	Base clause	Target clause
cause/3	cause/3	why_attracts	why_attracts
gravity/3	opposite_sign/3	force_gravity	opposite_charge
greater/3	greater/3	major_temp	major_mass
attraction/3	attraction/3	attracts	attracts
revolve_around/3	revolve_around/3	temp_planet	mass_e
inanimate/1	inanimate/1	temp_sun	mass_n
mass/2	charge/2	revolve	revolve
		planet	electron
		mass_planet	q_electron
		mass_sun	q_nucleus
		sun	nucleus

Applying the mapping engine presented above, we obtained the results in Table 3. [17] shows that temperature difference between sun and planets and mass difference between electrons and nucleus are noise, giving evidence that the mapping (*mass/2, mass/2*), as presented in [2], is wrong. Our mapping strategy is able to avoid the traps due to relations having the same name but that are not analogous in the considered context. Indeed, the only relation that has not been mapped compared to the outcome of STAR-2 is the noisy one. This analogy reveals that the reason why the electron revolves around the nucleus is already expressed in the atom description, since the cause of the attraction is the opposite charge, recognized as being analogous to the difference in masses between the sun and the planets in the Solar system.

Based on the analogy just seen, our cross-domain generalization (the meta-pattern) is:

pattern(solar_system(sun,planet), rutherford_atom(nucleus,electron)) :–
inanimate(sun_OR_nucleus), inanimate(planet_OR_electron),
mass_OR_charge(sun_OR_nucleus,mass_sun_OR_q_nucleus),
mass_OR_charge(planet_OR_electron,mass_planet_OR_q_electron),
attracts(sun_OR_nucleus,planet_OR_electron,attracts),
revolve_around(planet_OR_electron,sun_OR_nucleus,revolve),
greater(temp_sun_OR_mass_n,temp_planet_OR_mass_e,
major_temp_OR_major_mass),
gravity_OR_opposite_sign(mass_sun_OR_q_nucleus,
mass_planet_OR_q_electron,force_gravity_OR_opposite_charge),
cause(force_gravity_OR_opposite_charge,attracts,why_attracts).

The expected explanation of the physical phenomenon is, more or less, that "the difference in masses, together with the mutual attraction of the nucleus and the election, causes the electron to revolve around the nucleus". The inference hypotheses from *Solar system* to *Rutherford atom* are:

1: *greater(q_nucleus, q_electron, skolem_major_temp)*
2: *skolem_attraction_mass(major_mass, attracts, skolem_major_mass_attracts)*
3: *cause(skolem_major_mass_attracts, revolve, skolem_cause_revolve)*
4: *skolem_gravity(mass_n, mass_e, major_charge)*

where 1, 2 and 3 fully satisfy this interpretation. Additionally, hypothesis 4, obtained by transposing the gravity to the electromagnetic force, explains that there is a force based on the charge of the particles.

5 Conclusions

Analogy is a fundamental inference mechanism to transpose knowledge from known to unknown domains, producing new conclusions that are helpful to solve a problem. In addition to using equal descriptors, as in the mainstream literature, our approach can map also different descriptors that play the same role in the two domains. This paper has shown an approach by which the found analogies can be generalized into meta-patterns, that represent core knowledge and allow further reasoning. This enables more complex reasoning, since by finding an analogy between a meta-pattern and a novel description one may recognize analogies across several different stories, each having a different domain. This can be viewed more in general as a multi-strategy reasoning approach, in which analogies yield generalizations, that in turn are used as novel descriptions to reason over and over again.

Compared to the current literature, our approach allows to learn patterns representing the intuition that leads to a potential solution to the problem, and provides a computational trick that allows to reuse analogies computed in the past. Moreover, it can capture non-syntactic alignments without using meta-descriptions. The challenge to which this approach aims at contributing is the integration of the learned generalizations in the general knowledge network that an agent builds over its lifetime. Such an objective, aimed at overcoming a limit in the current landscape, is not trivial, since *integration* means defining strategies for knowledge addition and retrieval. Even harder difficulties are present in knowledge modification and deletion, since these tasks refer to incremental learning, which is still an open issue.

Future improvements will regard the recognition and mapping of relations with opposite sense. Another interesting direction will be the use of a probabilistic approach to assess the reliability of the mappings. In a multi-strategy perspective, we will study the use of an abductive procedure to check whether the inferred knowledge (mapped or projected) is consistent with the constraints of the target domain, and of an abstraction operator that shifts the representation when needed. Moreover, it will be interesting to allow the analogical reasoner to take advantage of available common sense knowledge, in order to check the soundness of the final result.

References

1. Leonidas A. A. Doumas, John E. Hummel, and Catherine M. Sandhofer. A theory of the discovery and predication of relational concepts. *Psychological Review*, 115(1):1–43, 2008.
2. Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63, 1989.
3. Dedre Gentner. Analogy. *A companion to cognitive science*, pages 107–113, 1998.
4. Dedre Gentner and Arthur B. Markman. Structure mapping in analogy and similarity. *American psychologist*, 52:45–56, 1997.
5. Attilio Giordana, Lorenza Saitta, and Davide Roverso. Abstracting concepts with inverse resolution. In *8th International Workshop on Machine Learning*, pages 142–146, 1991.
6. Douglas R. Hofstadter and Melanie Mitchell. The copycat project: A model of mental fluidity and analogy-making. In *Advances in Connectionist and Neural Computation Theory*. Ablex Publishing Corporation, Norwood, NJ, 1994.
7. Keith J. Holyoak and John E. Hummel. The proper treatment of symbols in a connectionist architecture. In E. Dietrich and A. Markman, editors, *Cognitive Dynamics: Conceptual and Representational Change in Humans and Machines.* Lawrence Erlbaum Associates, Mahwah, NJ, 2000.
8. Keith J. Holyoak and John E. Hummel. Understanding analogy within a biological symbol system. In Keith J. Holyoak Dedre Gentner and Boicho N. Konikov, editors, *The analogical mind*, pages 161–195. The MIT Press, Cambridge, MA, 2001.
9. Keith J. Holyoak and Paul Thagard. Analogical mapping by constraint satisfaction. *Cognitive Science*, 13:295–355, 1989.
10. John E. Hummel and Keith J. Holyoak. Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review*, 104(3):427–466, 1997.
11. Fabio Leuzzi and Stefano Ferilli. Reasoning by analogy using past experiences. In *Proceedings of the 28th Italian Conference on Computational Logic (CILC 2013)*, volume 1068, pages 115–129. CEUR-WS.org, 2013.
12. John W. Lloyd. *Foundations of Logic Programming, 2nd Edition*. Springer, 1987.
13. Hongjing Lu, Dawn Chen, and Keith J. Holyoak. Bayesian analogy with relational tansformations. *Psychological Review*, 119(3):617–648, 2012.
14. R. S. Michalski. Inferential theory of learning: Developing foundations for multistrategy learning. In *Machine Learning: A Multi-strategy approach*, volume 4, pages 3–62. Morgan Kaufmann Publishers, 1993.
15. D. E. Rumelhart, P. Smolensky, J. L. McClelland, and G. E. Hinton. Parallel distributed processing. chapter Schemata and Sequential Thought Processes in PDP Models, pages 7–57. MIT Press, Cambridge, MA, USA, 1986.
16. Manuela M. Veloso and Jaime G. Carbonell. Derivational analogy in prodigy: Automating case acquisition, storage, and utilization. In *Machine Learning*, pages 249–278, Boston, 1993. Kluwer Academic Publishers.
17. William H. Wilson, Graeme S. Halford, Brett Gray, and Steven Phillips. The star-2 model for mapping hierarchically structured analogs. *The analogical mind*, pages 125–159, 2001.

Spectral Features for Audio Based Vehicle Identification

Alicja Wieczorkowska[1], Elżbieta Kubera[2], Tomasz Słowik[3], and Krzysztof Skrzypiec[4]

[1] Polish-Japanese Academy of Information Technology,
Koszykowa 86, 02-008 Warsaw, Poland
alicja@poljap.edu.pl
[2] University of Life Sciences in Lublin, Department of Applied Mathematics and Computer Science, Akademicka 13, 20-950 Lublin, Poland
elzbieta.kubera@up.lublin.pl
[3] University of Life Sciences in Lublin, Department of Energetics and Transportation
Akademicka 13, 20-950 Lublin, Poland
tomasz.slowik@up.lublin.pl
[4] Maria Curie-Skłodowska University in Lublin,
Pl. Marii Curie-Skłodowskiej 5, 20-031 Lublin, Poland
krzysztof.skrzypiec@poczta.umcs.lublin.pl

Abstract. In this paper we address automatic vehicle identification based on audio information. Such data are complicated, as they depend on vehicle type, tires, speed and its change. In our previous research we designed a feature set for selected vehicle classes, discriminating pairs of classes. Now, we decided to expand the feature vector and find the best feature set (mainly based on spectral descriptors), possibly representative for each investigated vehicle category, which can be applied to a bigger data set, with more classes. The paper also shows problems related to vehicles classification, which is detailed in official documents by national authority for issues related to the national road system, but simplified for automatic identification purposes. Experiments on audio-based vehicle type identification are presented and conclusions are shown.

Keywords: Intelligent Transport System, Vehicle classification, Audio Signal Analysis

1 Introduction

The traffic we experience every day in the roads generates a lot of noise. Many countries measure this traffic and monitor its density. Such monitoring generates data that can be later analyzed, in order to estimate how the roads are used, introduce noise prevention etc. The audio data from the traffic monitoring are the subject of research presented in this paper. The reason is that audio data require less storage space, are cheaper to obtain, and can be recorded at night or at other low visibility conditions, for instance during bad weather, etc. They are easier to install, also in a way that is not visible for the drivers, so they are

less distractive. In the case of video cameras, the drivers are expecting a radar device, and change their behavior, so audio only recording can be even preferred. Still, extracting information from the audio data is not simple.

Audio data representing vehicles passing by are very complex, as they depend on many factors. The noise generated by vehicles depends on the vehicle type, speed, traffic intensity, how old the vehicles are, technical parameters, engine type, tires, exhaust system, air intake system, and other factors [12]. If different vehicles have the same type of engine, they sound very similar. On the other hand, the same vehicle sounds different when traveling upwards, downwards, with uniform speed or accelerating/decelerating. Also, the noise generated by old vehicles in very bad condition will be raised by few dB. Diesel engine is up to 5 dB louder than gasoline engine, whereas electric motor produces very little noise. At very low speed, below 30 km/h, electric motors are hardly audible. This actually is dangerous for pedestrians, as in this case they do not hear the vehicle approaching. At higher speed, tire friction makes these vehicles audible. Also, the road surface is an important factor of vehicle noise, and the difference can be about 5 dB or more.

In order to assess the road traffic in Poland, measurements are performed on various designated roads, at specified dates through the observed year, in day time (6am - 10pm), at night (10pm - 6am), and additionally between 8am and 4pm for trucks. The measurements are taken through week days, on Saturdays, Sundays and holidays [8]. Measurements can be done automatically, semi-automatically, or manually. In other countries, data about traffic are also collected. European Union also issued a directive on the framework for the deployment of Intelligent Transport Systems [5], with the purpose (among others) of the facilitation of the electronic data exchange between urban control centers for public or private transport. The United States also prepared a strategic plan for Intelligent Transport Systems (ITS) [11].

1.1 Related Work

The research on audio-based automatic classification of vehicles has already been performed, for varying number of classes. Such a research is usually performed for low sampling rate, 8-11.025 kHz, or downsampled for faster processing, and the analyzing window is usually short, 10-50ms. In our research, we decided to use 48 kHz/24 bit recordings, as this is the standard in modern audio recorders. Also, we decided to use longer analyzing frame, 330ms, to have high resolution spectrum, and longer frames yielded better results in our previous research [14].

Various classifiers have already been applied for audio-based vehicle classification, often with feature selection; extensive literature review on this subject is presented in [7]. In [9], artificial neural network was applied for 3 car classes (and horn as 4th class). Erb himself applied SVM (support vector machines) and feature selection with linear prediction for 3 classes: car, truck, and van. He obtained 87% correctness for vehicles traveling at low speed, and 83% for higher vehicle speeds. For traffic without given probabilities, the best result reached 80%, and increased to 83% if class probabilities matched those from the training

data [7]. Alexandre et al. in [1] applied multi-layer perceptrons combined with feature selection based on a genetic algorithm, for another 3 classes: car, motorcycle, and truck. Features included mel-frequency cepstral coefficients (MFCC), and zero crossing rate, yielding 93% correctness for 22 features and 75% for 66 features [1]. Four target classes were investigated in [15]: bus, car, motor, and truck. The authors used quadratic and linear discriminant analysis, and also k-nearest neighbors method (k-NN) and support vector machines (SVM). Feature vector included, among others, short time energy, average zero cross rate, and pitch frequency of periodic segments of signals, yielding 80% correctness for SVM with 12 Mel coefficients [15]. Generally, such research usually aims at recognizing 3-4 classes, for various vehicles, including military ones (see [6]).

1.2 Vehicle Classes

The vehicles can be classified in various ways. In Poland, according to the General Directorate for National Roads and Motorways, the vehicles are classified into the following classes: bicycles, motorcycles (including scooters), cars (including minibuses), vans (light trucks, up to 3.5t), small trucks (above 3.5t), big trucks (above 3.5t with trailers or semitrailers), buses, and tractors (including rollers, excavators etc.) [8]. Detailed specification is also prepared for tax and customs purposes [13]. Modern vehicle classification techniques that can be used for vehicle type recognition are based on data sets of vehicle outlines. However, similar vehicles can vary in the noise they generate.

All these categories include various types vehicles, differing in the noise produced. For instance, scooters differ from motorcycles with respect to the noise generated. Also, cars include vehicles for up to 9 passengers, including the driver. Off-road vehicle fall into this category, and they can produce more noise if all terrain tires are used. Emergency vehicle also produce different sounds when using audible warning devices. In order to obtain relatively uniform representation of each class, we decided to use most typical vehicles for the following 7 classes:

1. bus,
2. small truck (without trailers),
3. big truck - tractor unit with semitrailer,
4. van,
5. motorcycle, excluding scooters,
6. car,
7. tractor.

Minibuses, scooters, and emergency vehicles using audible warning devices were excluded from our research. Also bicycles were excluded, as they produce almost no sound (we recorded several examples). Tractor units without trailers or semitrailers were also excluded.

Our data. In this paper, we address automatic identification of vehicle type based on audio signal for 7 target classes: bus, small truck, big truck, van, motorcycle, car, and tractor. The audio and video recordings were made in a suburban area near Lublin in November 2012 (tractors) and June 2015 (other vehicles). The position of the audio recorder is shown in Figure 1. The road is approximately flat and straight here. The video material was used to mark ground truth data, whereas audio data was used for further investigations. Our goal was to parameterize the audio data for automatic recognition of vehicle type.

Fig. 1. The position of data acquisition

2 Feature Set

Based on our previous research [14], our features are based on 330ms audio segments, Hamming windowed for spectrum calculation. Most of the features are spectral, plus zero crossing rate (temporal feature). The feature vector includes standard features used in audio classification, plus additional features designed to discern objects representing our target classes.

The features applied are listed below:

- *Audio Spectrum Envelope* - 33 features, SE0, ..., SE32 [17],
- *SUM_SE* - sum of the spectrum envelope values,
- *MAX_SE_V, MAX_SE_IND* - value/index of spectrum envelope maximum,
- *F0_ACor, F0_MLA* - fundamental frequency calculated from the autocorrelation function, and through maximum likelihood algorithm [20],
- *EnAb4kHz* - proportion of the spectral energy above 4kHz to the entire spectrum energy;

- *Energy* - energy of the entire spectrum;
- *Audio Spectrum Centroid* (SC) - the power weighted average of the frequency bins in the power spectrum. Coefficients were scaled to an octave scale anchored at 1 kHz [17];
- *Audio Spectrum Spread* (SS) - RMS (root mean square) of the deviation of the log frequency power spectrum wrt. *Audio Spectrum Centroid* [17];
- *Zero Crossing Rate* (ZCR) in the time-domain of the sound wave; a zero-crossing is a point where the sign of the function changes;
- *RollOff* - the frequency below which 85% (experimentally chosen threshold) of the accumulated magnitudes of the spectrum is concentrated,
- *A14, A41, A15, A51, A16, A61, A17, A71, A24, A42, A52, A26, A62, A72, A34, A43, A35, A53, A63, A73, A45, A54, A47, A74, A56, A65, A57, A75, A67, A76* - normalized (with respect to the spectrum energy) energies Axy in the spectral ranges determined in such a way that the energy of this frequency range separates classes x and y, i.e. the class x shows higher energy values than the class y in this range; detailed ranges are shown in Table 1. Not for all pairs of classes such discerning ranges were found;
- *B14, B15, B16, B17, B24, B26, B34, B35, B45, B47, B56, B57, B67* proportion of energies between the indicated spectral ranges, Bxy=Axy/Ayx;
- *BW_10dB, BW_20dB, BW_30dB* - bandwidth of the frequency band comprising the spectrum maximum (in dB scale) and the level drop by 10, 20 and 20 dB, respectively, towards both lower and upper frequencies,
- *f_bus, f_smallTruck, f_bigTruck, f_van, f_motorcycle, f_car, f_tractor* - features discerning a particular class from all other classes, obtained through multiplication of all available Bxy values; the value for the target class should exceed those for other classes (at least this is the case for the data used to determine the frequency ranges Axy).

Altogether, the feature set consists of 97 features. Some of these features were used in our previous research on vehicle classification [14]. New features added in this paper include *Audio Spectrum Envelope*, *SUM_SE*, *MAX_SE_V*, *MAX_SE_IND*, *F0_ACor, F0_MLA*, BW_10dB, BW_20dB, BW_30dB, and *f_bus, f_smallTruck, f_bigTruck, f_van, f_motorcycle, f_car, f_tractor*. Features Axy and Bxy were calculated in the same way as in [14] but for 7 classes and for different audio samples (only tractor samples were the same).

The data for experiments were recorded in stereo; the average of both channels was used for calculating features. For each audio frame, the obtained feature vector can be used as input to classifiers, to identify vehicle(s) audible in this frame. Using binary classifiers allows recognition of plural vehicles per segment, i.e. all recognized target class (for instance, big truck and car).

2.1 Feature Selection

After designing the feature set and performing experiments on this set, we also applied feature selection, as our feature vector is relatively large, so such a procedure is recommended in this case [10]. For each of the classifiers investigated,

Table 1. Spectral ranges Axy: the energy of this frequency range separates classes x and y, i.e. the class x shows higher energy values than the class y in this range

Axy	lower limit [Hz]	upper limit [Hz]
A14	790.8203125	723.6328125
A15	17.578125	38.0859375
A16	72.9296875	796.875
A17	26.3671875	46.875
A41	1798.828125	1822.265625
A51	3275.390625	3278.320313
A61	937.5	1054.6875
A71	3369.140625	3418.945313
A24	32.2265625	1116.210938
A26	23.4375	750
A42	1986.328125	2071.289063
A52	117.1875	290.0390625
A62	1001.953125	1297.851563
A72	4209.960938	4212.890625
A34	49.8046875	1183.59375
A35	383.7890625	1127.929688
A43	2554.6875	2589.84375
A53	117.1875	375
A63	4283.203125	4309.570313
A73	3843.75	3849.609375
A45	732.421875	1125
A47	691.40625	1374.023438
A54	117.1875	571.2890625
A74	298.828125	325.1953125
A56	111.328125	541.9921875
A57	87.890625	137.6953125
A65	1069.335938	1397.460938
A75	867.1875	896.484375
A67	770.5078125	1403.320313
A76	316.40625	515.625

3-fold cross validation was applied. We tested 2 versions: with constant number of features to be selected (10 features; number arbitrarily chosen), and with feature importance above a selected threshold (0.5 mean decrease of Gini criterion; threshold arbitrarily chosen, based on the observation of feature importance for all classes). Since better results were obtained in the second case, we decided to choose this feature selection scheme.

3 Experiments

In our experiments we applied SVM, random forests (RF, [3]), and deep learning (DL) architecture (neural network), see Section 3.1, using R and packages: h2o, randomForest, and e1071 [16], [18]. In each case, we trained a binary classifier

for each target class, to recognize automatically whether a target vehicle sound is present in the analyzed audio data (positive answer of the classifier) or not (negative answer). This is because multiple vehicles can be recorded in the same audio sample, and such samples represent multi-label data. A set of binary classifiers can perform multi-label classification, identifying each vehicle present in the analyzed audio sample.

3.1 Classifiers

SVM looks for a decision surface (hyperplane) that maximizes the margin around the decision boundary. The decision hyperplane should be maximally away from the training data points, called support vectors. Data that is not linearly separable is projected into a higher dimensional space where it is linearly separable. This mapping is done by using kernel functions. In our case, we used kernels in form of radial basis functions (RBF). Such a function has 2 parameters, c and γ, which require tuning for best performance. We applied automatic tuning available in R package (tune.svm).

RF is a set of decision trees, constructed with minimizing bias and correlations between the trees. Each tree is built without pruning to the largest possible extent, using a different N-element bootstrap sample of the N-element training set, i.e. obtained through drawing with replacement. For a K-element feature vector, k features are randomly selected ($k \ll K$, often $k = \sqrt{K}$) for each node of any tree. The best split on these k features is used to split the data in the node, and Gini impurity criterion is minimized to choose the split. The Gini criterion measures of how often an element would be incorrectly labeled if labeled randomly, according to the distribution of labels in the subset. This procedure is repeated M times, to obtain M trees; M=500 in our experiments (standard setting in R). Classification of is performed by simple voting of all trees in RF.

DL architecture is composed of multiple levels of non-linear operations. DL neural network architecture is a multi-layer neural net, with many hidden layers. This algorithm is implemented in h2o as feedforward neural net, with automatic data standardization. Training is performed through back propagation, with adaptive learning. Weights are iteratively updated in so-called epochs, with grid-search of the parameter space. H2o parameters include large weight penalization and drop-out regularization (ignoring a random fraction of neuron inputs). Standard setting of DL in h2o were used in our experiments.

3.2 Data and Results

The data used in our experiments represented 21 frames of positive examples for bus class, 26 for small truck, 39 for big truck, 33 for van, 15 for motorcycle, 33 for car, and 18 for tractor. Negative examples outnumbered the positive for each class, as this reflects the real situation. The data were divided into training and testing part (approximately 2/3 for training and 1/3 for testing), with different vehicles data used for training and for testing. No cross validation was applied in the initial experiments. The audio data represented sound of a single vehicle, or

multiple vehicles. Positive examples contained sounds of the target class (possibly accompanied with other sounds), and negative examples represented any other classes (single or multiple vehicles), or silence.

The error for our data is shown in Figure 2. As we can see, the error is usually small, with the highest error for small truck classification using RF (but still much better than random choice). Even though the error in this particular case was quite dissatisfying, RF are still useful, as we can use them to estimate the importance of the proposed features. As examples, we present importance for car, motorcycle and tractor in Figure 3. MeanDecreaseGini used here is a measure of feature importance based on the Gini impurity index used for the calculation of splits during RF training. When a split of a node is made, Gini index for the two descendent nodes G1 and G2 is less than for the parent node, G0, and the importance I is calculated as I=G0-G1-G2.

As we can see, our proposed features are of high importance in these cases. Other important features are related to spectral envelope, also in plots not presented in this paper.

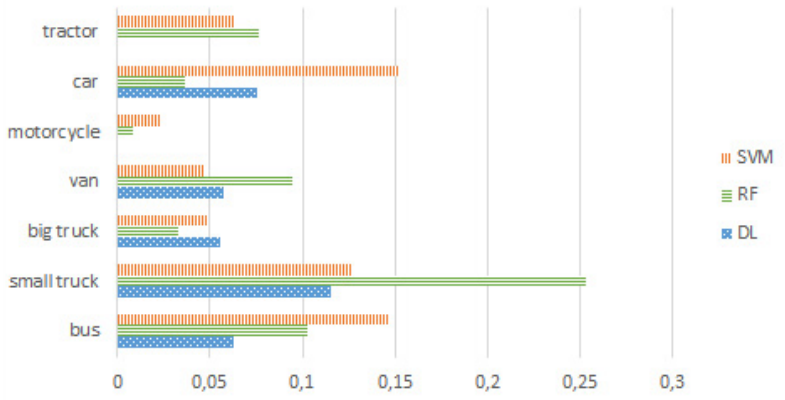


Fig. 2. Classification error

For illustration purposes, the details of true positive (tp) recognition, true negative (tn), false positive (fp) and false negative (fn) for SVM is shown in Table 2. As we can see, there are fp and fn, but not so many, and we should remember that video data can be also used together with audio for better classification.

We also performed clustering experiments, in order to check how the proposed feature set is grouping vehicle objects [19]. Clustering into 7 clusters was performed. Exemplary clustering is presented in Table 3. As we can see, data are a bit mixed in clusters. For example, tractor samples are together with motorcycle data, which can be surprising. Still, usually most of the objects are located in one cluster. Only cars and vans are together in one cluster, but these vehicles are similar with respect to produced sound anyway. The obtained clustering

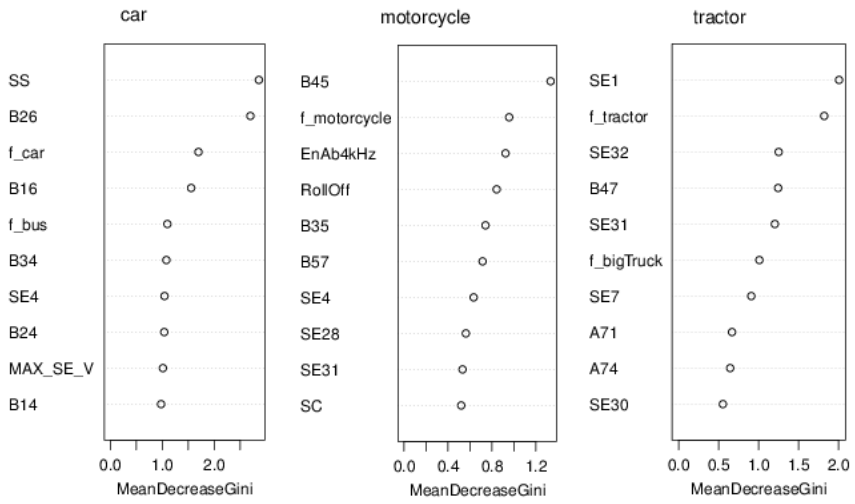


Fig. 3. Importance plot for car, motorcycle, and tractor classes

Table 2. True positive (tp) recognition, true negative (tn), false positive (fp) and false negative (fn) for SVM classifier

vehicle	tp	tn	fp	fn
bus	0	41	0	7
small truck	4	72	3	8
big truck	7	132	1	6
van	6	95	0	5
motorcycle	1	124	0	3
car	3	42	1	7
tractor	1	73	0	5

shows that our feature set describes vehicle sounds quite well, and explains good results of the classification performed on these data. We should also remember that audio data depend on many factors, including tires, speed, acceleration etc., and these factors can be additionally investigated in further research.

Since our data are imbalanced, we also decided to balance the data. This can be done through downsampling the negative examples for each binary classifiers, or upsampling the examples of each target class [4]. In our experiments, we decided to perform upsampling, i.e. replicating the target class frames. After balancing, equal number of positive and negative examples (audio frames) for each classifier were obtained, i.e. 48-146. Classification was performed in 3-fold cross-validation. Next, we performed feature selection, as mentioned before, again in 3-fold cross validation for each classifier. Features of importance exceeding 0.5 threshold of mean decrease of Gini index were kept in the final feature set. The following features were present in each fold for the target classes:

Table 3. Hierarchical Ward's clustering with Euclidean metrics for our data

cluster no.:	1	2	3	4	5	6	7
bus	12	1	0	0	0	0	0
small truck	3	3	1	0	3	0	3
big truck	4	7	2	0	0	0	0
van	0	0	0	0	0	10	0
motorcycle	0	0	0	2	6	0	0
car	0	0	0	0	0	10	0
tractor	0	0	0	10	8	0	0

- bus: SE6-8, SE10-11, SE21-23, SE25, SC, A14, A15, A51, A16, A17, A71, B17, A24, A26, A34, A35, A73, A54, B45, A74, B47, A56, A57, B67, f_bus;
- small truck: SE0, SE4, SE7, SE11, SE14-16, SE30, SUM_SE, MAX_SE_V, Energy, A14, A16, A61, A24, A52, A26, A34, A35, A53, B35, A45, A54, B45, A47, A56, A67, A76
- big truck: SE0-1, SE3, SE5, SE9-15, SUM_SE, F0_Acor, A14, A16, A61, A24, B24, A52, A26, A34, A35, A53, A63, A45, A54, A47, A74, B47, A56, A57, A75, A67, A76;
- van: SE0, SE7-11, SE13, SE16, SE20, SE23, SE26-32, SUM_SE, F0_Acor, EnAb4kHz, SC, SS, ZCR, RollOff, A14, A41, A15, A16, A61, B16, A17, A24, A52, A26, A62, B26, A34, A43, A53, A54, B45, A74, B47, A56, A65, B56, A67, A76, f_bus, f_smallTruck, f_bigTruck, f_van, f_motorcycle, f_car;
- motorcycle: SE1, SE4, SE6-7, SE13, SE15, Se22, SE25-32, EnAb4kHz, RollOff, B15, A61, A52, B35, A45, B45, A47, B47, A75, f_bigTruck, f_motocycle;
- car: SE19-21, SE29-30, SUM_SE, MAX_SE_V, BW_30dB, F0_Acor, SC, SS, ZCR, B14, B16, B24, A52, B26, A43, B34, A53, B56, f_smallTruck, f_bigTruck, f_van, f_car;
- tractor: SE1, SE7, SE9, SE22-24, SE28-29, SE31-32, A71, B17, A73, B45, A74, B47, f_bigTruck, f_motorcycle, f_tractor.

As we can see, the feature designed to identify the target classes, or to discern between pairs of classes, are of high importance and are kept in the feature set after the feature selection procedure.

These experiments were performed for the balanced data. Classification error and F-measure after feature selection are shown in Figures 4 and 5, respectively. As we can see, classification error decreased after feature selection, and deep learning classifiers yields best results.

4 Summary and Conclusions

The features proposed in this paper for audio-based classification of vehicle type yields good results, as the error is below 10-15% in most cases and improves after feature selection, for 7 classes, which compares favorably with other research, performed for 3-4 classes for similar data. Best results were obtained for deep learning neural network. Still, our results can be improved, and we hope to

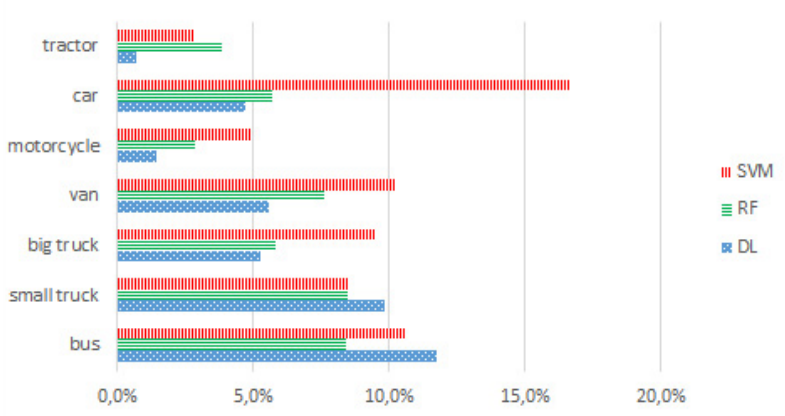


Fig. 4. Classification error after feature selection

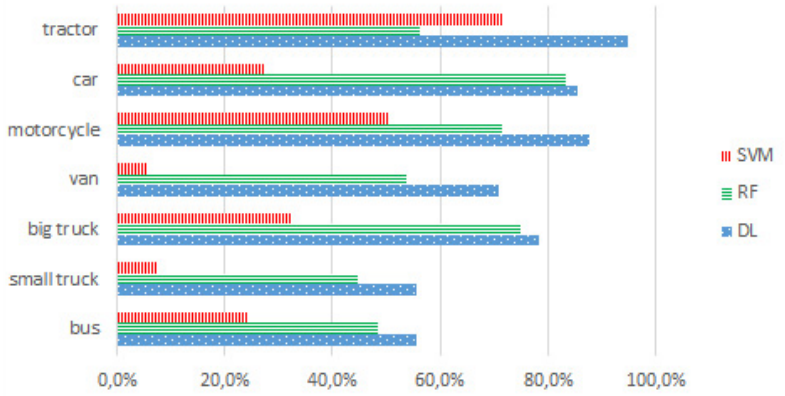


Fig. 5. F-measure after feature selection

get better results when more data are collected. Also, taking Doppler effect into account may further improve the results, see [2], where the data were compared with prerecorded sounds. We can also include subclasses not investigated in this research (scooters, emergency vehicles etc.), and perform hierarchical classification, as it usually improves the results [14]. Other factors than vehicle type can also be taken into account, including vehicle speed, acceleration, tires, etc. Also, video material can be used together with audio data in the vehicle classification task.

Acknowledgments. This work was partially supported by the Research Center of PJAIT, supported by the Ministry of Science and Higher Education in Poland.

References

1. Alexandre, E., Cuadra, L., Salcedo-Sanz, S., Pastor-Sánchez, A., Casanova-Mateo, C.: Hybridizing Extreme Learning Machines and Genetic Algorithms to Select Acoustic Features in Vehicle Classification Applications. Neurocomputing 152, 58–68 (2015)
2. Berdnikova, J., Ruuben, T., Kozevnikov, V., Astapov, S.: Acoustic Noise Pattern Detection and Identification Method in Doppler System. Elektronika ir Elektrotechnika 18(8), 65–68 (2012)
3. Breiman, L.: Random Forests. Machine Learning 45, pp. 5–32 (2001); see also: http://www.stat.berkeley.edu/~breiman/RandomForests/cc_papers.htm
4. Chen, C., Liaw, A., Breiman, L.: Using Random Forest to Learn Imbalanced Data, http://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf
5. Directive 2010/40/Eu of the European Parliament and of the Council of 7 July 2010 on the framework for the deployment of Intelligent Transport Systems in the field of road transport and for interfaces with other modes of transport, http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2010:207:0001:0013:EN:PDF
6. Duarte, M.F., Hu, Y.H.: Vehicle classification in Distributed Sensor Networks. J. Parallel Distrib. Comput. 64, 826–838 (2004)
7. Erb, S.: Classification of Vehicles Based on Acoustic Features. Thesis, Graz University of Technology (2007)
8. General Directorate for National Roads and Motorways (GDDKiA, in Polish) https://www.gddkia.gov.pl/userfiles/articles/z/zarzadzenia-generalnego-dyrektor_13901/zarzadzenie\%2038\%20Wytyczne\%20-\%20Zalacznik\%20d\%20-\%20Instrukcja\%20\%20GPR_2015.pdf
9. George, J., Cyril, A., Koshy, B.I., Mary, L: Exploring Sound Signature for Vehicle Detection and Classification Using ANN. International Journal on Soft Computing 4(2) 2013
10. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Data Mining, Inference, and Prediction. Springer Series in Statistics, Springer (2009)
11. ITS 2015–2019 Strategic Plan, http://www.its.dot.gov/strategicplan.pdf
12. Iwao, K., Yamazaki, I.: A Study on the Mechanism of Tire/Road Noise. JSAE Review 17, 139–144 (1996)
13. Izba Celna w Przemyślu (the Customs Chamber in Przemyśl, in Polish) http://www.przemysl.ic.gov.pl/download/sprowadzauto/zas_klasyfikacji_pojazdow_samo.pdf
14. Kubera, E., Wieczorkowska, A., Skrzypiec, K.: Audio-Based Hierarchic Vehicle Classification for Intelligent Transportation Systems. Submitted to ISMIS 2015. Springer, LNAI (2015).
15. Mayvan, A.D., Beheshti, S.A., Masoom, M.H.: Classification of Vehicles Based on Audio Signals using Quadratic Discriminant Analysis and High Energy Feature Vectors. International Journal on Soft Computing 6, 53–64 (2015)
16. Package 'h2o', http://cran.r-project.org/web/packages/h2o/h2o.pdf
17. The Moving Picture Experts Group, http://mpeg.chiariglione.org/standards/mpeg-7
18. The R Foundation, http://www.R-project.org
19. Struyf, A., Hubert, M., Rousseeuw, P.J.: Clustering in an Object-Oriented Environment, http://www.jstatsoft.org/v01/i04/paper
20. Zhang, X, Marasek, K., Raś, Z.W.: Maximum Likelihood Study for Sound Pattern Separation and Recognition. 2007 International Conference on Multimedia and Ubiquitous Engineering MUE 2007, IEEE, 807–812 (2007)

Probabilistic Frequent Subtree Kernels

Pascal Welke[1], Tamás Horváth[1,2], and Stefan Wrobel[2,1]

[1] Dept. of Computer Science, University of Bonn, Germany
[2] Fraunhofer IAIS, Schloss Birlinghoven, Sankt Augustin, Germany

Abstract. We propose a new probabilistic graph kernel defined by the set of frequent subtrees generated from a polynomial size random sample of spanning trees of the transaction graphs. In contrast to other frequent pattern graph kernels, it can be computed in polynomial time for any arbitrary graphs. Due to its probabilistic nature, the embedding function corresponding to our graph kernel is not always correct. Our empirical results on chemical graph datasets, however, clearly demonstrate that the graph kernel we propose is much faster than other frequent pattern based graph kernels, with only marginal loss in predictive accuracy.

1 Introduction

Over the past decade, graph kernels (see, e.g., [6]) have become a well-established approach in graph mining for their excellent predictive performance. One of the early graph kernels, the *frequent subgraph kernel*, is based on explicitly embedding the graphs into a feature space spanned by the set of all frequent connected subgraphs in the input graph database [5]. It was empirically demonstrated in [5] that remarkable predictive accuracies can be obtained with this type of graph kernels on the molecular graphs of small pharmacological compounds.

One of the main drawbacks of frequent subgraph kernels is that the preprocessing step of generating *all* frequent connected subgraphs from the input database is computationally intractable [8]. Many of the practical approaches ignore this limitation, implying that such systems can be infeasible even for small datasets. Approaches that do not disregard the computational limitation above resort either to various heuristics for traversing the search space that result in incomplete output (see, e.g., [3]), or restrict the class of the input graphs to some tractable class to guarantee both completeness and efficiency (see, e.g., [9]).

In this work we propose a new approach different from the ones above. On the one hand, we are interested in a frequent subgraph kernel that is not restricted to any particular graph class. On the other hand, we require the kernel to be efficiently computable. It follows from the negative complexity result [8] on mining frequent connected subgraphs from arbitrary graphs that, unless P = NP, such a frequent subgraph kernel can only be achieved by relaxing the conditions in some way. For this work, we give up the demand on completeness and 'represent' each input graph by a random sample of spanning trees that is of polynomial size. Such a random sample is always a forest and can be generated in polynomial time. Combining these facts with the positive result that frequent

subtree mining in forests can be solved with polynomial delay (see, e.g., [2,9]), we arrive at a polynomial delay frequent subgraph mining algorithm.

Our approach is sound, but incomplete in two ways: First, it is only able to identify frequent subtrees and not arbitrary graph patterns. Second, even if a tree pattern is frequent, it might not be identified as such by the algorithm. Another problem is the computational intractability of calculating the embedding for any (unseen) query graph. This step is based on the NP-complete problem of deciding for each frequent pattern if it is subgraph isomorphic to the query. Therefore, to compute the embedding for an unseen graph into the feature space, we follow the same probabilistic strategy as for the pattern mining step, i.e., generate a random subset of spanning trees of the graph and decide whether the tree pattern is subgraph isomorphic to any of the random spanning trees. Needless to say that subgraph isomorphism is decided with one-sided error in this way. This, somewhat unusual, idea is justified by the property that any tree found by our mining algorithm is not only frequent with respect to the database, but with high probability it has a relatively high frequency also in the set of spanning trees for each transaction graph containing it. Therefore, there is a high chance that the pattern will be detected with this method in the query graph as well, if it is part of it[3].

We have empirically evaluated the proposed approach on the NCI-HIV and on different subsets of the ZINC chemical datasets. The empirical results on these datasets show that at least 20% of *all* frequent subtrees can be recovered from a *single* random spanning tree per graph. Using one spanning tree per graph, we have observed a speed-up of at least a factor of three against frequent subgraph kernels that are defined by *all* frequent connected subgraphs. A more significant difference between the two approaches is that in contrast to the ordinary one, our method is able to process significantly larger datasets. In addition, as it generates a smaller feature set, it speeds up the subsequent learning step as well. It is natural to ask whether the feature set obtained by our method is expressive enough in terms of predictive performance compared to the original one. Our results on the NCI-HIV dataset show that there is only a marginal loss in predictive performance. This suggests that a careful composition of our simple technique with some other fast graph kernel might result in a fast graph kernel of high predictive performance.

The rest of the paper is organized as follows. In Section 2 we present our algorithm with some important implementation details. Section 3 describes the empirical evaluation of our approach and Section 4 concludes with some interesting questions for further work.

2 The Probabilistic Frequent Subtree Kernel

Several graph kernels have been developed over the past decade for predictive graph mining. A broad range of these graph kernels belong to the class of *con-*

[3] We assume that the query graph has been selected from the same (unknown) probability distribution as the graphs in the input database.

volution kernels [7]. That is, the input graphs are first decomposed into certain sets of substructures determined by some pattern language and the graph kernels are then defined by the intersection kernel over such sets. Depending on the particular choice of the substructure class (i.e., the pattern language), different graph kernels can be defined in this way. One of the first such graph kernels was defined by means of *frequent connected subgraphs* [5]. That is, the feature space corresponding to the kernel is spanned by the set of connected graphs that occur in at least a certain proportion of the graphs in the input database. The first step of this approach is to generate *all* frequent connected subgraphs, i.e., to solve the following pattern mining problem:

Frequent Connected Subgraph Mining (FCSM) Problem: *Given* a finite set $D \subseteq \mathcal{G}$ for some graph class $\mathcal{G}$ and a threshold $t \in (0, 1]$, *list* the set $F \subseteq \mathcal{P}$ of all pairwise non-isomorphic graphs from some graph class $\mathcal{P}$ that are subgraph isomorphic to at least $\lceil t \cdot |D| \rceil$ graphs in D.

In what follows, $\mathcal{G}$ and $\mathcal{P}$ will be referred to as *transaction* and *pattern* classes. The set F of frequent patterns in D naturally yields a binary vector representation for any arbitrary graph G: We map G to its characteristic vector $\overline{v}_G$ over the universe F, i.e., $\overline{v}_G$ is indexed by F and for all $H \in F$, $\overline{v}_G[H] = 1$ if and only if H is subgraph isomorphic to G. To avoid redundancies in the characteristic vectors over F, the patterns in F are required to be pairwise non-isomorphic.

One of the main limitations of graph kernels based on frequent subgraphs is the computational intractability of the FCSM problem: If there is no restriction on the transaction graphs in $\mathcal{G}$ and $\mathcal{P}$ consists of all connected graphs of $\mathcal{G}$ then, unless P = NP, the FCSM problem cannot be solved in output polynomial time [8]. Unfortunately, this negative complexity result is reflected in practice even for sparse graphs (see, e.g., [10]), making the frequent subgraph kernel infeasible for arbitrary graph datasets. One way to avoid this limitation is to restrict $\mathcal{G}$ to some graph class for which the FCSM problem can be solved at least in incremental polynomial time, preferably, with guaranteed polynomial delay. Examples of such classes include the class of bounded tree-width graphs [9] and some of its proper subclasses [10,13]. Our goal is, however, to propose a frequent pattern based graph kernel that can be calculated in time polynomial in the cardinality of frequent patterns and in the size of the input graphs G_1, G_2 for any *arbitrary* graphs G_1 and G_2. Note that other 'tree kernels' (e.g. [11,12]) do not directly relate to our work, as they (i) use homomorphism instead of subgraph isomorphism as the embedding operator and (ii) are not frequency based.

2.1 Probabilistic Frequent Subtrees

For the reason mentioned above, we focus on restrictions of the pattern class $\mathcal{P}$. Though most state-of-the-art frequent subgraph mining algorithms require the transaction graph class $\mathcal{G}$ and the pattern class $\mathcal{P}$ to be equal (up to connectivity), this constraint is not necessary and can therefore be dropped. In this work

we allow the pattern and transaction graph classes to differ and focus on mining frequent subtrees in databases of arbitrary graphs. For the sake of simplicity, we assume that the transaction graphs are connected by noting that the mining algorithm presented in this work can naturally be generalized to disconnected transaction graphs as well.

Restricting the pattern language to trees alone, however, is not sufficient to get rid of the computational intractability mentioned above; mining frequent trees in arbitrary graphs is not possible in output polynomial time, as it could otherwise be used to decide the Hamiltonian path problem in polynomial time [8]. There are some transaction graph classes for which this problem can be solved in incremental polynomial time, or even with polynomial delay (e.g., for transaction graphs of bounded tree-width [9] or for cactus graphs with bounded cycle degree [13]). Since, however, our goal is a kernel for arbitrary transaction graphs, we cannot resort to these approaches.

In light of the above mentioned negative results on mining frequent subtrees in arbitrary graphs, we give up the demand on *completeness* and consider only such subsets of the set of all frequent subtrees that can be enumerated with polynomial delay. A natural candidate towards this direction is to generate first some random subset of the spanning trees for each transaction graph and, in a second step, to enumerate the set of frequent subtrees from the forest database obtained in this way.

With this problem relaxation we arrive at an easy to implement and, as shown in Section 3, practically effective frequent subgraph mining algorithm (see Algorithm 1). In addition to the transaction database D and the frequency threshold t given in the definition of the FCSM problem, the input contains an additional parameter $k \in \mathbb{N}$ defining an upper bound on the number of spanning trees to be generated for each transaction graph. The algorithm starts by sampling (no more than) k spanning trees for each graph in the database. Instead of mining frequent patterns in the input database D directly, we represent each graph G by a forest F_G formed by the vertex disjoint union of the random spanning trees generated for G. This effectively reduces the problem of mining frequent subtrees in arbitrary graph databases D to the problem of mining frequent subtrees in a database D' consisting of forest transactions F_G for all $G \in D$. A tree T is said to be t-frequent in such a setting if and only if it is subgraph isomorphic to at least $\lceil t \cdot |D'| \rceil = \lceil t \cdot |D| \rceil$ forests in D'. As frequent subtree mining in forest transaction databases can be done with polynomial delay [2,9], we arrive at an algorithm that runs in time polynomial in the combined size of D and the set of frequent subtrees in D'.

To distinguish between the output F of the frequent subgraph problem and the output F' of Algorithm 1 on D and t, we will refer to the former set as *frequent patterns* and to the later one as *probabilistic (subtree) patterns* with respect to a threshold t. Clearly, for any D, t, and k, the output of Algorithm 1 is a subset of the set of frequent trees in D, i.e., Algorithm 1 is sound. However, it will not necessarily find all frequent patterns, i.e., it is not complete in general. Thus, with this technique, on the one hand we obtain a polynomial time

Given: A graph database $D \subseteq \mathcal{G}$ an integer $k > 0$ and an integer threshold $t > 0$.
Output: A set of t-frequent subtrees of D.

1: $D' := \emptyset$
2: **for all** $G \in D$ **do**
3: Sample k spanning trees of G uniformly at random
4: Add the forest of those trees up to isomorphism to D'
5: List all t-frequent subgraphs in D'

Algorithm 1: The Probabilistic Subtree Mining Algorithm

algorithm that is fast for small values of k, on the other hand, however, loose a number of frequent patterns.

Given a set F' of probabilistic tree patterns and an unseen query graph G, the embedding of G into the feature space spanned by F' cannot be computed in polynomial time (if P $\neq$ NP). This is because deciding subgraph isomorphism from a tree into an arbitrary graph is NP-complete. We can, however, use the same probabilistic embedding based on a random sample of spanning trees of G as for the transaction graphs. Clearly, this embedding is not unique because it depends on the random sample of the spanning trees generated for G. It can, however, be shown that with high probability it has a small Hamming distance to the exact one defined by F'.

In the application context of graph kernels, the incompleteness of Algorithm 1 raises two important questions:

1. How stable is the output of Algorithm 1 and what is its recall with respect to *all* frequent subtrees?
2. How does the incompleteness of our algorithm influence the predictive performance of the graph kernel obtained?

Regarding the first question, we show in the next section on real-world chemical datasets that (i) the output is very stable even for $k = 1$ and (ii) more than 75% of the frequent patterns can be recovered by using only ten random spanning trees per graph (i.e., for $k = 10$). Regarding the second question, we show on the NCI-HIV benchmark dataset that our experimental results are comparable with those obtained by the FSG algorithm [5]. Before presenting these and other empirical results in Section 3, we first discuss some implementation issues and analyse the time complexity of Algorithm 1.

2.2 Implementation Issues and Runtime Analysis

Line 3 of Algorithm 1 can be implemented using Wilson's algorithm [14], which has an expected runtime that is linear in the *mean hitting time* of a graph and returns each spanning tree of G with the same probability. This is $O\left(n^3\right)$ in the worst case, but conjectured to be much smaller for most graphs [14]. The set of all sampled spanning trees *up to isomorphism* (Line 4) can be computed from the set of sampled spanning trees using some canonical string representation

for trees and a prefix tree as data structure (see, e.g., [2] for more details on canonical string representations for labeled graphs). We follow this approach to practically reduce the runtime of the subsequent frequent subtree mining step, as isomorphic spanning trees yield the same subtrees and can safely be omitted. For each tree, this can be done in $O(n \log n)$ time by computing first the tree center and then applying a canonical string algorithm for rooted trees as in [2]. These canonical strings are then stored in and retrieved from a prefix tree in time linear in their size.

Thus, the sampling step of our algorithm runs in expected $O\left(kn^3\right)$ time. If we do not require the spanning trees to be drawn uniformly, we can improve on this time and achieve a deterministic $O(km \log n)$ runtime, where m denotes the number of edges. This is achieved by choosing a *random* permutation of the edge set of a graph and then applying Kruskal's minimum spanning tree algorithm using this edge order. It is not difficult to see that this technique can generate random spanning trees with non-uniform probability. As our experimential results on molecular graphs of pharmacological compounds show, this has no significant impact on the predictive performance of the graph kernel obtained.

Finally we note that for Line 5, we can use almost any one of the existing algorithms generating frequent connected subgraphs (i.e., subtrees) from forest databases (see, e.g., [2] for an overview on this topic).

3 Experiments

In this section we empirically evaluate our proposed method on the NCI-HIV benchmark and on the ZINC molecular graph datasets. We start with the description of these datasets.

NCI-HIV consists of $42,687$ compounds of which $39,337$ are connected[4]. The average number of vertices and edges per graph are 41 and 43, respectively. The transactions are annotated with their activity against the human immunodeficiency virus (HIV). The molecules are labeled by "active" (A), "moderately active" (M), or "inactive" (I). We consider the following three usual binary classification problems:

(i) A and M together versus I (AMvsI),
(ii) A versus M and I (AvsMI), and
(iii) A versus I where instances labeled M are removed (AvsI).

ZINC is a subset of $8,946,757$ ($8,946,755$ connected) so called 'Lead-Like' molecules from the zinc database of purchasable chemical compounds[5]. The molecules in this subset have a molar mass between $250g/mol$ and $350g/mol$ and have an average number of vertices and edges 43 and 44, respectively.

[4] `http://cactus.nci.nih.gov/`
[5] `http://zinc.docking.org/subsets/lead-like`

		$k=1$	$k=2$	$k=3$	$k=10$
NCI-HIV	$t=5\%$	0.2013 ± 0.0120	0.3553 ± 0.0134	0.4648 ± 0.0051	0.7832 ± 0.0085
	$t=10\%$	0.2026 ± 0.0222	0.3445 ± 0.0142	0.4540 ± 0.0159	0.7994 ± 0.0182
	$t=20\%$	0.2445 ± 0.0138	0.3976 ± 0.0168	0.5041 ± 0.0114	0.8338 ± 0.0140
ZINC	$t=5\%$	0.3680 ± 0.0087	0.5670 ± 0.0165	0.6842 ± 0.0094	0.9250 ± 0.0045
	$t=10\%$	0.3277 ± 0.0189	0.5136 ± 0.0184	0.6447 ± 0.0140	0.9249 ± 0.0118
	$t=20\%$	0.3103 ± 0.0259	0.4899 ± 0.0305	0.6141 ± 0.0341	0.9053 ± 0.0128

Table 1. Recall with standard deviation of the probabilistic tree patterns on the NCI-HIV and ZINC datasets for frequency thresholds 5%, 10%, and 20%

For each dataset, we only consider the compounds that have a connected graph and use the molecular graph representations that include hydrogen atoms.

To generate frequent subgraph patterns, we used FSG [5]. In Line 5 of our algorithm, we also used FSG to generate frequent subtrees. In this way, we can consistently compare the runtimes of the two methods, as none of them is affected by some specific heuristic not used in the other one. However, we expect a significant improvement of our probabilistic method over the tradidtional one, once a specialized tree mining algorithm is applied. All our experiments were conducted on an Intel i7 CPU with 3.40GHz and 16GB of RAM running Ubuntu 14.04. Our algorithms were implemented in C and compiled using gcc.

3.1 Recall of the Probabilistic Subtree Pattern Space

As discussed in Section 2, for any graph database, the pattern set F' found by our algorithm is a subset of all frequent subtrees F_T, which in turn is a subset of all frequent subgraphs F. We now analyze the recall of our method, i.e. the amount of frequent subtree patterns that are found when applying Algorithm 1 for various k and t. To this end, let $R(k,t) := \frac{|F'|}{|F_T|}$ be the fraction of t-frequent tree patterns that are found if Algorithm 1 selects k random spanning trees.

For each dataset, we sample 10 subsets of 100 graphs each and report the average value of $R(k,t)$ and its standard deviation. Using the FSG algorithm, on each subset we first compute all frequent connected patterns, including non-tree patterns as well, and then filter out all frequent subgraphs that are not trees. We have found that at least 95% of all frequent subgraphs are trees. In a second step, we apply Algorithm 1 to the subset and divide the size of its output by the size of the frequent subtrees computed previously. The results on the two datasets can be found in Table 1 for different values of k with frequency thresholds 5%, 10%, and 20%. It can be seen that the fraction of the retrieved tree patterns rapidly grows with the number of sampled spanning trees per graph. Sampling 10 spanning trees per graph already results in around 90% recall for the ZINC dataset and in a recall of 80% for the NCI-HIV dataset.

Iteration	1	2	3	4	5	6	7	8	9	10
NCI-HIV	3920	20	5	10	14	7	2	6	7	2
ZINC	9898	18	17	11	10	22	7	7	9	1

Table 2. Repetitions of the experiment with $k = 1$ sampled trees. The numbers reported are the number of probabilistic patterns that were not in the union of all probabilistic patterns found up to the current iteration.

3.2 Stability of Probabilistic Subtree Patterns

Section 3.1 indicates that, at least on pharmacological molecular graphs, a relatively high recall of the frequent tree patterns can be achieved, even for a very small number of random spanning trees. In this section we report empirical results showing that even for $k = 1$, the output pattern set of Algorithm 1 is quite stable. To empirically demonstrate this advantageous property, we have run Algorithm 1 10 times for the same values of the parameters k and t and observed how the union of the probabilistic tree patterns grows.

To this end, we fix two sets of graphs, each of size approximately $40,000$, as follows: We take all connected graphs in NCI-HIV, as well as a random subset ZINC_{40k} of ZINC that contains $40,000$ graphs. We run Algorithm 1 ten times for the datasets obtained with parameters $k = 1$ and $t = 10\%$. Each execution results in a set F'_i of probabilistic subtree patterns, from which we define $U_i = \bigcup_{j=0}^{i} F'_j$ with $F'_0 := \emptyset$. Table 2 reports $|F'_i \setminus U_{i-1}|$, i.e., the number of *new* probabilistic subtree patterns found in iteration i for $i = 1, \ldots, 10$. For an initial number of $3,920$ (NCI-HIV) and $9,898$ (ZINC_{40k}) probabilistic patterns, the number of newly discovered patterns drops to at most 22 for the following iterations. This shows, that the feature set generated does not depend too much on the spanning trees selected at random. Overall, we have found that independent runs of our algorithm yield similar feature sets on the same data.

3.3 Comparison of Recall and Stability Experiments

It can be seen from Tables 1 and 2 that the number of patterns found by repeating the sampling twice for $k = 1$ is much lower than that found by a single repetition of the sampling for $k = 2$. This is not surprising, considering the notion of "frequency" we use.

Indeed, suppose we "represent" a graph database D by another database D_k obtained by choosing k *non-isomorphic* spanning trees for each $G \in D$. Then the set F'_k of frequent tree patterns in D_k is equal to the union of the probabilistic patterns for the implicitly $k^{|D|}$ tree datasets, where each set is obtained by choosing one out of the k spanning trees for each graph. Suppose a tree T is t-frequent for some $t \in (0, 1)$ in D_k. Then there are at least $\lceil t \cdot |D_k| \rceil$ forests in D_k that contain T, implying that there is at least one tree in each such forest that contains T. Therefore, choosing those trees and an arbitrary tree from each forest that does not contain T results in a database of trees where T is t-frequent.

	$\leq n$	$\leq n\sqrt{n}$	$\leq n^2$
NCI-HIV	34.36%	63.35%	86.29%
ZINC	45.84%	89.63%	99.62%

Table 3. Distribution of the graphs in NCI-HIV and ZINC with respect to different upper bounds on their number of spanning trees.

The other direction needed for proving the equality of the two pattern sets is trivial. Thus, by selecting at least $k = 2$ random spanning trees per graph, we exponentially blow up the number of combinations to be considered, in contrast to the strategy of repeating Algorithm 1 polynomially many times for $k = 1$. As long as k is bounded by a polynomial in the size of D, the algorithm runs in time polynomial in the combined size of D and F'_k.

Although the exponential "blow-up" of the combinations increases the chance of finding a frequent tree pattern, the number of all combinations is doubly exponential because a graph may have exponentially many spanning trees in general. The impressive experimental results presented in this section for relatively small values of k can be explained by the fact that most molecular graphs used in our experiments have quadratically many spanning trees only, as shown in Table 3.

Finally we note that if we sample k spanning trees per graph, we might get less than k pairwise non-isomorphic spanning trees because of collisions (e.g. there is only one spanning tree, if the graph itself is already a tree). Thus $k^{|D|}$ is only an upper bound on the number of one-spanning-tree-per-graph databases that can be generated from D_k.

3.4 Predictive Performance

Using the annotated dataset NCI-HIV, in this section we show that the predictive performance of the probabilistic subtree kernel compares favorably with that of the frequent subgraph kernel. We choose, as does most related work, a wrapper method and report the achieved area under the ROC-curve (AUC) of a well trained support vector machine (SVM) [4]. To this end, we consider the three binary classification problems described at the beginning of this section. We compare the predictive performance of (i) the frequent subgraph kernel computed by FSG [5] with that of (ii) the probabilistic frequent subtree kernel for $k = 1$ and $k = 2$, and for frequency thresholds of 5%, 10%, 15%, and 20%. For (ii), we use only the results with Wilson's random spanning tree sampling algorithm; we obtained nearly identical accuracy and runtime results with the greedy sampling algorithm based on Kruskal's method for all combinations of the above parameters. For our evaluation, we use the SVM provided by the libSVM package [1] with a radial basis function kernel.

We repeat Algorithm 1 four times using different sets of sampled trees and report the average and standard deviation of AUC values from a 3-fold cross validation for each resulting feature set. The same procedure is applied to the frequent subgraph pattern set, here we use a different splitting for the cross validation in each run.

t		AvsI	AMvsI	AvsMI
5%	Frequent Patterns	o.o.m	o.o.m	o.o.m
	Uniform Trees $k=1$	0.8927 ± 0.002	0.7235 ± 0.0023	0.8823 ± 0.0024
	Uniform Trees $k=2$	0.8994 ± 0.0012	0.7409 ± 0.0069	0.8909 ± 0.0074
10%	Frequent Patterns	0.9131 ± 0.0038	0.7529 ± 0.0024	0.9082 ± 0.0031
	Uniform Trees $k=1$	0.8853 ± 0.0081	0.7132 ± 0.0054	0.8745 ± 0.0118
	Uniform Trees $k=2$	0.8828 ± 0.0151	0.7109 ± 0.0021	0.8729 ± 0.0062
15%	Frequent Patterns	0.9134 ± 0.0026	0.7488 ± 0.0013	0.9093 ± 0.0031
	Uniform Trees $k=1$	0.8772 ± 0.0075	0.7062 ± 0.0055	0.8676 ± 0.0071
	Uniform Trees $k=2$	0.8762 ± 0.0071	0.7108 ± 0.0051	0.8676 ± 0.0042
20%	Frequent Patterns	0.9135 ± 0.0039	0.7424 ± 0.0026	0.9057 ± 0.0017
	Uniform Trees $k=1$	0.8675 ± 0.0076	0.6855 ± 0.0073	0.86 ± 0.0074
	Uniform Trees $k=2$	0.864 ± 0.01	0.6879 ± 0.0061	0.8579 ± 0.0074

Table 4. Average AUC values for the three learning problems on the NCI-HIV benchmark dataset for the frequent subgraph kernel and the probabilistic frequent subtree kernel for $k = 1, 2$ and for different frequency thresholds.

The results are presented in Table 4. On the one hand, one can see that from a frequency threshold of 10%, the results with the frequent subgraph kernel are more stable than those with the probabilistic frequent subtree kernel on all three problems. Though the frequent subgraph kernel outperforms the probabilistic frequent subtree kernel on the same frequency threshold, the difference seems marginal once we compare the best results on each problem, especially in light of the runtime presented in the next section. On the other hand, however, for the frequent subgraph kernel, the results could be calculated only for $t = 10\%$, while for the probabilistic frequent subtree kernel we obtained the result in half of the time for $t = 5\%$. For this frequency threshold, we were unable to produce any result because the system ran out of memory. In fact, we had difficulties training the SVM using all frequent patterns, even for larger frequency thresholds, because of its excessive memory usage. These observations clearly show the limitation of the frequent subgraph kernel over the probabilistic frequent subtree kernel when the predictive performance required can be achieved only for low frequency thresholds. Finally we note that there is no improvement when sampling two instead of one spanning tree per graph. Therefore, in the next section we concentrate on sampling a single spanning tree per graph.

3.5 Speedup

In this section, we compare the runtime of FSG and our algorithm (using FSG as the mining subroutine) on different subsets of the ZINC dataset. We use Wilson's method to generate a random spanning tree for each graph in all our experiments and report the combined time for sampling and frequent pattern generation. We used the implementation of the FSG algorithm [5] that is pro-

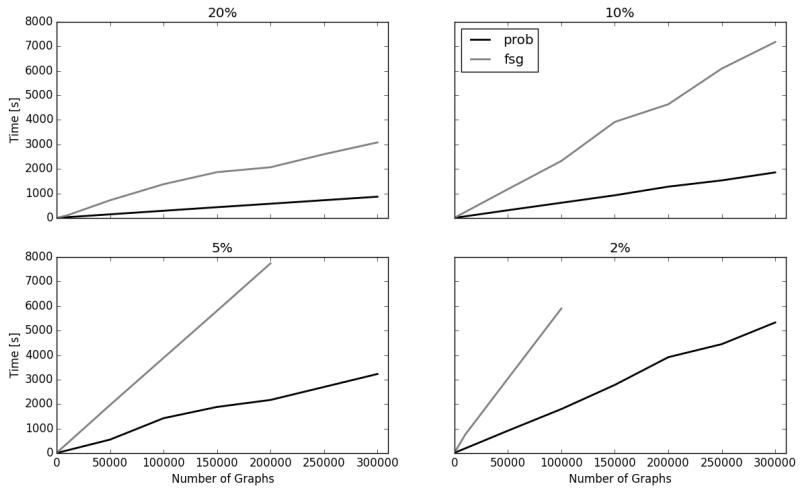


Fig. 1. Runtime results in seconds for our method (black) and FSG (grey), for different frequency thresholds. The x-values show the number of graphs in the subsets of ZINC that were used.

vided by its authors[6]. As already noted, one could use a specialized frequent subtree mining algorithm in combination with our sampling method to further increase the speedup. We experimented with several such publicly available tree mining algorithms but, somewhat surprisingly, they were not able to beat the speed of FSG on the tree data set.

Figure 1 shows the time in seconds for our algorithm with $k = 1$ in black and for FSG in gray. It reports results with subsets of the ZINC dataset of increasing size. One can see that our method outperforms FSG on all datasets and for all frequency thresholds. Furthermore, with decreasing frequency threshold, our method has an increasing absolute speed-up. Last but not least, in contrast to our method, FSG has a clear limitation beyond $200,000$ graphs for $t = 5\%$ and beyond $100,000$ graphs for $t = 2\%$.

4 Conclusion and Future Work

We have presented a kernel for graph structured data that is based on probabilistic subtree patterns, i.e., on frequent subtrees in a forest database obtained by randomly selecting a polynomially bounded set of spanning trees for each transaction graph in the input database. Our empirical results on the NCI-HIV benchmark graph dataset show that the probabilistic feature space considered is expressive enough compared to that of frequent subgraph kernels when applying

[6] `http://glaros.dtc.umn.edu/gkhome/pafi/overview`

them to predictive graph mining. This advantageous property even holds if we sample only a single spanning tree from each graph. Furthermore, our graph kernel is not only faster than the frequent subgraph kernel, but has a much smaller memory footprint in all stages.

We are currently working on some formal properties of the proposed method, e.g., on probabilistic guarantees for (t_1, t_2)-frequent subtrees where t_1 is a frequency threshold for a tree pattern within the set of spanning trees of a graph, whereas t_2 within the database. These results will then be turned into an algorithm that, for a given confidence value δ specified by the user, generates each frequent subtree with probability at least δ. We are also considering the design and implementation of a frequent subtree mining algorithm for unlabeled free trees that is able to effectively process massive forest transaction datasets.

One of the strengths of our method is that it is not restricted to any particular graph class. This advantageous property allows us to empirically investigate the proposed graph kernel on more complicated graph classes beyond molecular graphs, such as the k-neighborhood graphs of the web graph or RDF graphs.

References

1. C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.
2. Y. Chi, R. R. Muntz, S. Nijssen, and J. N. Kok. Frequent subtree mining - an overview. *Fundam. Inform.*, 66(1-2):161–198, 2001.
3. D. J. Cook and L. B. Holder. Substructure discovery using minimum description length and background knowledge. *J. Artif. Intell. Res. (JAIR)*, 1:231–255, 1994.
4. C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
5. M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis. Frequent substructure-based approaches for classifying chemical compounds. *Knowledge and Data Engineering, IEEE Transactions on*, 17(8):1036–1050, 2005.
6. T. Gärtner. *Kernels for structured data*. PhD thesis, Universität Bonn, 2005.
7. D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, Univerisity of California - Santa Cruz, July 1999.
8. T. Horváth, B. Bringmann, and L. De Raedt. Frequent hypergraph mining. In *Inductive Logic Programming*, pages 244–259. Springer, 2007.
9. T. Horváth and J. Ramon. Efficient frequent connected subgraph mining in graphs of bounded tree-width. *Theor. Comput. Sci.*, 411(31-33):2784–2797, 2010.
10. T. Horváth, J. Ramon, and S. Wrobel. Frequent subgraph mining in outerplanar graphs. *Data Min. Knowl. Discov.*, 21(3):472–508, 2010.
11. P. Mahé and J.-P. Vert. Graph kernels based on tree patterns for molecules. *Machine learning*, 75(1):3–35, 2009.
12. N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *The Journal of Machine Learning Research*, 12:2539–2561, 2011.
13. P. Welke, T. Horváth, and S. Wrobel. On the complexity of frequent subtree mining in very simple structures. *Proceedings of Inductive Logic Prigramming 2014*, 2015.
14. D. B. Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 296–303. ACM, 1996.

Heterogeneous networks decomposition and weighting with text Recompile mining heuristics

Jan Kralj[1,2], Marko Robnik-Šikonja[3] and Nada Lavrač[1,2,4]

[1] Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
[2] Jožef Stefan International Postgraduate School, Jamova 39, 1000 Ljubljana, Slovenia
[3] Faculty of Computer and Information Science, Večna pot 113, 1000 Ljubljana, Slovenia
[4] University of Nova Gorica, Vipavska 13, 5000 Nova Gorica, Slovenia

Abstract. The paper presents an approach to mining heterogeneous information networks applied to a task of categorizing customers linked in a heterogeneous network of products, categories and customers. We propose a two step methodology to classify the customers. In the first step, the heterogeneous network is decomposed into several homogeneous networks by using different connecting nodes. Similarly to the construction of bag-of-words vectors in text mining, we assign larger weights to more important nodes. In the second step, the resulting homogeneous networks are used to classify data either by network propositionalization or label propagation. Because the data set is highly imbalanced we adapt the label propagation algorithm to handle imbalanced data. We perform a series of experiments and compare different heuristics used in the first step of the methodology, as well as different classifiers which can be used in the second step of the methodology.

Keywords: · Network analysis · Heterogeneous information networks · Network decomposition · PageRank · Text mining heuristics · Centroid classifier · SVM

1 Introduction

The field of *network analysis* is well established and exists as an independent research discipline since the late seventies [24] and early eighties [1]. In recent years, analysis of *heterogeneous information networks* [20] has gained popularity. In contrast to standard (homogeneous) information networks, heterogeneous networks describe heterogeneous types of entities and different types of relations.

This paper addresses the task of mining *heterogeneous information networks* [20]. In particular, for a user-defined node type, we use the method of classifying network nodes through network decomposition; this results in homogeneous networks whose links are derived from the original network. Following [6], the method constructs homogeneous networks whose links are weighed by counting the number of intermediary nodes, connecting two nodes. After the individual

homogeneous networks are constructed, we consider two approaches for classification of network nodes. We classify the nodes either through label propagation [25], or using a propositionalization approach [6], which allows the use of standard classifiers such as the centroid and SVM classifier on the derived feature vector representation. The propositionalization approach was already applied to a large heterogeneous network of scientific papers from the field of psychology in our previous work [13]. In this work, we propose two improvements to the presented methodology: 1) we propose a new variant of label propagation algorithm which improves classification performance on imbalanced data sets; 2) we introduce heuristics for homogeneous network construction, inspired by the word weighting used in text mining and information retrieval.

The paper is structured as follows. Section 2 presents the related work. Section 3 presents the two-stage methodology for classification in heterogeneous networks. We present the method for constructing homogeneous networks from heterogeneous networks and two methods for classification of nodes in a network: a network propositionalization technique and a label propagation algorithm. Section 4 presents how this method can be further improved. We first introduce a variant of the label propagation algorithm with improves performance on imbalanced data sets. We then show how the homogeneous network construction can be improved by using different weighting heuristics. Sect. 5 presents the application of the methodology on a challenge data set of customers linked to products they purchased, followed by the three stage analysis of the results. First, we examine the effect of using different classifiers in a final step of classification via propositionalization. Second, we test different heuristics for homogeneous network construction. Finally, we analyze the improved label propagation method for imbalanced data sets. Section 6 concludes the paper and presents plans for further work.

2 Related work

In data analysis for networks, instances are connected in a network of connections. In ranking methods like Hubs and Authorities (HITS) [11], PageRank [18], SimRank [9] and diffusion kernels [12], authority is propagated via network edges to discover high ranking nodes in the network. Sun and Han [20] introduced the concept of *authority* ranking for heterogeneous networks with two node types (bipartite networks) to simultaneously rank nodes of both types. Sun et al. [21] address authority ranking of all nodes types in heterogeneous networks with a star network schema, while Grčar et al. [6] apply the PageRank algorithm to find PageRank values of only particular type of nodes.

In network classification, a typical task is to find class labels for some of the nodes in the network using known class labels of remaining network nodes. A common approach is propagation of labels in the network, a concept used in [25] and [23]. An alternative approach to label propagation, namely classification of network nodes through propositionalization, is described in [6]. There, a heterogeneous network is decomposed into several homogeneous networks which are

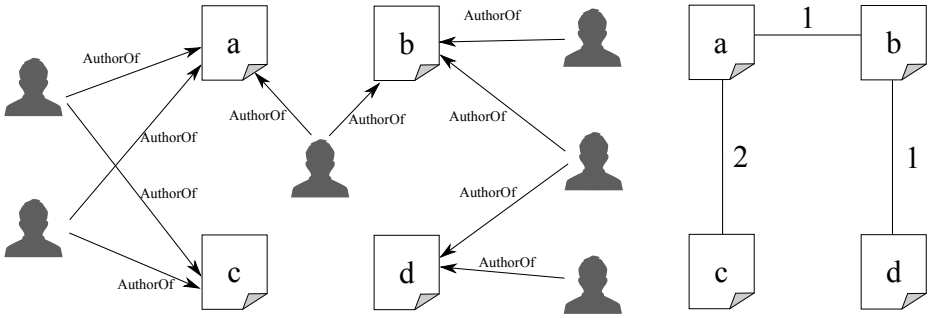


Fig. 1. An example of a heterogeneous network, decomposed into a homogeneous network where papers are connected if they share a common author. The weight of the edges is equal to the number of authors that contributed to both papers

used to create feature vectors corresponding to nodes in the network. The feature vectors are used by a SVM [16, 14, 4], kNN [22] or centroid classifier [7] to predict class values of these nodes. The network propositionalization approach was also used in [13].

Our work is also related to text mining, specifically to bag-of-words vector construction. Here it is important to correctly set weights of terms in documents. Simple methods like term frequency are rarely used, as the term-frequency inverse-document-frequency (tf-idf) weighting introduced in [10] is more efficient. A number of weighting heuristics which also take into account labels of documents have been proposed, such as the χ^2, the information gain [3], the Δ-idf [17], and the relevance frequency [15].

3 Methodology

We first present a two step methodology to mine heterogeneous information networks (defined by Sun and Han [20]), in which a certain type of nodes (called the target type) is labeled. In the first step of the methodology, the heterogeneous network is decomposed into a set of homogeneous networks. In the second step, the homogeneous networks are used to predict the labels of target nodes.

3.1 Network decomposition

The original heterogeneous information network is firt decomposed into a set of homogeneous networks, containing only the target nodes of the original network. In each homogeneous network two nodes are connected if they share a particular direct or indirect link in the original heterogeneous network. Take as an example a network containing two types of nodes, *P*apers and *A*uthors, and two edge types, *Cites* (linking papers to papers) and *Written_by* (linking papers to authors). From it, we can construct two homogeneous networks of papers: the first, in which two papers are connected if one paper cites another, and the second, in

which they are connected if they share a common author (shown in Fig. 1). The choice of links used in the network decomposition step requires expert who takes the meaning of links into account and chooses only the decompositions relevant for a given task.

3.2 Classification

In the second step of the methodology, the homogeneous networks are used to classify the nodes. We compare two approaches to this task: the label propagation algorithm [25] and the network propositionalization approach [6].

Label propagation. The label propagation algorithm starts with a network adjacency matrix $M \in \mathbb{R}^{n,n}$ and a class matrix $Y \in \mathbb{R}^{n,|C|}$, where $C = \{c_1, \ldots, c_m\}$ is the set of classes, with which the network nodes are labeled. The j-th column of Y represents the j-th label of C, meaning that Y_{ij} is equal to 1 if the i-th node belongs to the j-th class and 0 otherwise. The algorithm constructs the matrix $S = D^{-\frac{1}{2}} M D^{-\frac{1}{2}}$, where D is a diagonal matrix and the value of each diagonal element is the sum of the corresponding row of M. The algorithm iteratively computes $F(t) = \alpha S F(t-1) + (1-\alpha)Y$ until there are no changes in the matrix $F(t)$. The resulting matrix F is used to predict the class labels of all unlabeled nodes in the network. Zhou et al. [25] show that the iteration converges to the same value regardless of the starting point $F(0)$. They also show that the value F^* that $F(t)$ converges to can also be calculated by solving a system of linear equations, as

$$F^* = (I - \alpha S)^{-1} Y. \tag{1}$$

To classify a heterogeneous network, decomposed into k homogeneous networks, we propose classification of nodes using *all* available connections from all k homogeneous network. We construct a new network with the same set of nodes as in the original homogeneous networks. The weight of a link between two nodes is calculated as the sum of link weights in all homogeneous networks. In effect, if the original networks are represented by adjacency matrices $M_1, M_2, \ldots, M_k$, the new network's adjacency matrix equals $M_1 + M_2 + \cdots + M_k$.

Classification by propositionalization. An alternative method for classifying the target nodes in the original heterogeneous network (called network propositionalization) calculates feature vectors for each target node in the network. The vectors are calculated using the personalized PageRank (P-PR) algorithm [18]. The personalized PageRank of node v (P-PR$_v$) in a network is defined as the stationary distribution of the position of a random walker who starts its walk in node v and then at each node either selects one of the outgoing connections or travels to his starting location. The probability (denoted p) of continuing the walk is a parameter of the personalized PageRank algorithm and is usually set to 0.85. Once calculated, the resulting PageRank vectors are normalized according to the Euclidean norm. The vectors are used to classify the nodes from which they were calculated.

For a single homogeneous network, the propositionalization results in one feature vector per node. For classifying a heterogeneous network, decomposed into k homogeneous networks, Grčar et al. [6] propose to concatenate and assign weights to the k vectors, obtained from the k homogeneous networks. The weights are optimized using a computationally expensive differential evolution [19]. A simpler alternative is to use equal weights; due to the size of feature vectors in our experiments, we decided to follow this approach. Many classifiers, for example SVM classifier [16, 14, 4], kNN classifier [22] or a centroid classifier [7] can be used.

4 Methodology improvement

We present two improvements to the methodology described in Sect. 3. First, we describe handling of imbalanced data sets, then we present a novel edge weighting in construction of homogeneous networks from the original network.

4.1 Imbalanced data sets and label propagation

The label propagation approach may not perform well if the data is highly imbalanced, i.e., if the frequency of class labels are not approximately equal. We propose an adjustment of the label propagation algorithm i.e., to change the initial label matrix Y so that larger classes have less effect in the iterative process. The value of the label matrix Y in this case is no longer binary, but it is set to $\frac{1}{|c_j|}$ if node i belongs to class j and 0 otherwise.

If the data set is balanced (all class values are equally represented), then the matrix Y is equivalent to the original binary matrix multiplied by the inverse of the class value size. This, along with (1), means that the resulting prediction matrix only changes by a constant and the final predictions remain unchanged. However, if the data set is imbalanced, smaller classes have a greater effect in the iterative calculation of F^*. This prevents the votes of more frequent class values to outweigh less frequent class values votes.

4.2 Text mining inspired weights calculation

We recall weighting of terms in the construction of bag-of-words (BOW) vectors and explain how the same ideas can be applied to extraction of homogeneous networks from heterogeneous networks.

Term weighting in text mining. In bag-of-words vector construction one feature vector represents each document d in a corpus c of documents. In that vector, the i-th value corresponds to the i-th term (a word or a n-gram) that appears in the corpus. The value of the feature depends primarily on the frequency of the term in the particular document. We describe some methods for assigning the feature values. We use the following notations: $f(t, d)$ denotes the

number of times a term t appears in the document d and D denotes the corpus (a set of documents). We assume that the documents in the set are labeled, each document belonging to a class c from the set of all classes C. We use the notation $t \in d$ to describe that a term t appears in document d. Where used, the term $P(t)$ is the probability that a randomly selected document contains the term t, and $P(c)$ is the probability that a randomly selected document belongs to class c.

Scheme	Formula
`tf`	$f(t,d)$
`if-idf`	$f(t,d) \cdot \log\left(\frac{\lvert D\rvert}{\lvert\{d' \in D : t \in d'\}\rvert}\right)$
`chi^2`	$f(t,d) \cdot \sum_{c\in C}\left(\frac{(P(t\wedge c)P(\neg t\wedge\neg c) - P(t\wedge\neg c)P(\neg t\wedge c))^2}{P(t)P(\neg t)P(c)P(\neg c)}\right)$
`ig`	$f(t,d) \cdot \sum_{c\in C}\left(\sum_{c'\in\{c,\neg c\}}\left(\sum_{t'\in\{t,\neg t\}}\left(P(t',c') \cdot \log\frac{P(t'\wedge c')}{P(t')P(c')}\right)\right)\right)$
`delta-idf`	$f(t,d) \cdot \sum_{c\in C}\left(\log\frac{\lvert c\rvert}{\lvert\{d' \in D : d' \in c \wedge t \in d'\}\rvert} - \log\frac{\lvert\neg c\rvert}{\lvert\{d' \in D : d' \notin c \wedge t \notin d'\}\rvert}\right)$
`rf`	$f(t,d) \cdot \sum_{c\in C}\left(\log\left(2 + \frac{\lvert\{d' \in D : d' \in c \wedge t \in d'\}\rvert}{\lvert\{d' \in D : d' \notin c \wedge t \notin d'\}\rvert}\right)\right)$

Table 1. Weight heuristics in text mining

Table 1 shows different methods for term weighting. Term frequency (`tf`) weights each term with its frequency in the document. Term frequency–inverse document frequency (`tf-idf`) [10] addresses the drawback of the `tf` scheme, which tends to assign high values to common words that appear frequently in the corpus. The χ^2 (`chi^2`) weighting scheme Debole and Sebastiani [3] attempts to correct another drawback of the `tf` scheme (one which is not addressed by the `tf-idf` scheme) by taking also class value of processed documents into consideration. This allows the scheme to penalize terms that appear in documents of all classes, and favor terms which are specific to some classes. Information gain (`ig`) [3] uses class labels to improve term weights. The Δ-idf (`delta-idf`) [17] and the Relevance frequency (`rf`) [15] attempt to merge the ideas of `idf` and both above class-based schemes by penalizing both common and non-informative terms.

Midpoint weighting in homogeneous network construction. Let us revisit the example from Sect 3.1, in which two papers are connected by one link for each author they share. The resulting network is equivalent to a network in which two papers are connected by a link with a weight equal to the number of authors that wrote both papers (Fig. 1). The method treats all authors equally which may not be correct. For example, if two papers share an author that only co-authored a small number of papers, it is more likely that these two papers are similar than if the two papers share an author that co-authored tens or even hundreds of papers. The first pair of papers should therefore be connected by a stronger weight than the second. Moreover, if the papers are labeled by

the research field, then two papers, sharing an author publishing in only one research field, are more likely to be similar as if they share an author which has co-authored papers in several research fields. Again, the first pair of papers should be connected by an edge with larger weight.

Both described considerations are similar to the issues addressed in the term weighting schemes in document retrieval (presented at the beginning of this section). For example, the `tf-idf` weighting scheme attempts to decrease the weight of terms which appear in many documents, while we wish to decrease the weight of links, induced by authors which are connected to many papers. The `ig` weighting scheme decreases the weight of terms which appear in variously labeled documents, while we wish to decrease the weight of links induced by authors appearing in different research areas i.e., connected to variously labeled papers.

We alter the term weighting schemes in such a way that they can be used to set weights to midpoints in heterogeneous graphs (such as authors in our example). We propose that the weight of a link between two base nodes in the first step of the methodology (see Sect. 3) is calculated as the sum of weights of all the midpoints they share. In particular, if we construct a homogeneous network in which nodes are connected if they share a connection to a node of type T in the original heterogeneous network, then the weight of the link between nodes v and w should be equal to

$$\sum_{m \in T : (m,v) \in E \wedge (m,w) \in E} w(m), \tag{2}$$

where $w(m)$ is the weight assigned to the midpoint m. The value of $w(m)$ can be calculated in several ways. Table 2 shows the proposed midpoint weighting heuristics corresponding to term weighting used in document retrieval (Table 1).

Scheme	Formula
`tf`	1
`if-idf`	$\log\left(\frac{\lvert B\rvert}{\lvert\{b \in B : (b,m) \in E\}\rvert}\right)$
`chi^2`	$\sum_{c \in C} \frac{(P(m \wedge c)P(\neg m \wedge \neg c) - P(m,\neg c)P(\neg m, c))^2}{P(m)P(c)P(\neg m)P(\neg c)}$
`ig`	$\sum_{c \in C}\left(\sum_{c' \in \{c,\neg c\}}\left(\sum_{m' \in \{m,\neg m\}} P(m' \wedge c') \log\left(\frac{P(m' \wedge c')}{P(c')P(m')}\right)\right)\right)$
`delta-idf`	$\sum_{c \in C}\left\lvert \log \frac{\lvert c\rvert}{\lvert\{b' \in B : b' \in c \wedge (b',m) \in E\}\rvert} - \log \frac{\lvert\neg c\rvert}{\lvert\{b' \in B : b' \notin c \wedge (b',m) \notin E\}\rvert}\right\rvert$.
`rf`	$\sum_{c \in C}\left(\log\left(2 + \frac{\lvert\{b' \in B : b' \in c \wedge (b',m) \in E\}\rvert}{\lvert\{b' \in B : b' \notin c \wedge (b',m) \notin E\}\rvert}\right)\right)$

Table 2. Heuristics for weighting midpoints in homogeneous network construction.

The `tf` weight is effectively used in [6], where all authors are weighed equally. The `delta-idf` weighting scheme, unlike other term weighting schemes, may assign negative weights to certain terms. Since link weights in graphs are assumed

to be positive both by the PageRank and the link propagation algorithm, we must change the weighting scheme before it can be used to construct homogeneous networks. We propose that in the original weighting scheme, terms which receive negative values are also deemed informative, as they are informative about the term *not* being typical of a certain class. Therefore it is reasonable to take the absolute values of the weights in network construction.

5 Experimental setting and results

Data set description. We evaluated the proposed weighting heuristics on a data set of customer purchases used in the PAKDD 2015 mining competition *Gender prediction based on e-commerce data*. The data consists of $30,000$ customers. The data for each customer consists of the gender (the target variable), the start and end time of the purchase, and the list of products purchased. A typical product is described by a 4-part string (for example: `A3/B5/C2/D8`). The strings describe a 4-level hierarchy of products, meaning that the example product is the product $D8$ (or D-level category) which belongs to (A-level) category $A3$, sub-category (or B-level category) $B5$ and sub-subcategory (or C-level category) $C3$. The data set is highly imbalanced: $23,375$ customers are women and $6,625$ are men.

The data set implicitly defines a heterogeneous network consisting of five node types: customers (the base node type) and four hierarchy levels. We constructed four homogeneous networks from the original heterogeneous network. In the first, two customers are connected if they purchased the same product (same D-level item). In the second, they are connected if they purchased a product in the same sub-subcategory (C-level item), in the third if they purchased the same B-level item and in the fourth if they purchased the same A-level item. The constructed networks are referred to as A-, B-, C- and D-level networks from this point on.

Experiment description. The first set of experiments was designed to determine if the results of [6], which show that a centroid classifier, trained on Personalized PageRank feature vectors, performs as good as more complex SVM classifier. We tested the performance of the centroid classifier, the k-nearest neighbors classifier (with k set to $1, 2, 5$ and 10), and the SVM classifier. Because the data set is imbalanced we tested the SVM classifier both with uniform instance weights as well as weights proportional to the class frequencies. The tests were performed on feature vectors extracted from all four homogeneous networks. We randomly sampled $3,000$ network nodes to train all classifiers and tested their performance on the remaining $27,000$ nodes. The small size of the training set ensured that all classifiers trained fast.

In the second set of experiments we tested the heuristics, used in the construction of the homogeneous networks. We tested two classifiers: the SVM classifier which classified the network nodes using solely the Personalized PageRank vectors extracted from the network, and the label propagation classifier which

Classifier:	Centroid	1-nn	2-nn	5-nn	10-nn	SVM	SVM (balanced weights)
A-level network	74.19%	63.61%	71.93	72.74%	74.36%	74.03%	74.62%
B-level network	70.78%	56.42%	59.17%	65.30%	67.73%	63.51%	72.61%
C-level network	64.71%	63.62%	67.21%	68.26%	71.65%	70.15%	75.18%
D-level network	60.08%	67.36%	70.39%	66.72%	66.06%	65.61%	71.17%

(a) Results of the first set of experiments.

Scheme	A-level	B-level	C-level	D-level
`tf`	76.61%	74.00%	77.34%	73.65%
`chi^2`	77.80%	74.17%	76.86%	68.76%
`idf`	77.80%	74.22%	77.23%	72.25%
`delta`	77.80%	74.14%	77.23%	72.52%
`rf`	77.80%	74.11%	76.81%	70.54%
`ig`	77.80%	74.12%	76.87%	68.72%

(b) Performance of the SVM classifier in the second round of experiments.

Scheme	A-level	B-level	C-level	D-level
`tf`	75.52%	64.28%	63.60%	72.44%
`chi^2`	76.02%	65.15%	71.95%	72.75%
`idf`	74.90%	63.83%	61.02%	72.48%
`delta`	74.90%	63.76%	61.05%	72.48%
`rf`	75.52%	64.28%	67.59%	72.55%
`ig`	76.02%	65.15%	72.41%	72.96%

(c) Performance of the label propagation classifier in the second round of experiments.

Scheme	A-level	B-level	C-level	D-level
`tf`	77.16%	74.75%	77.28%	73.91%
`chi^2`	77.16%	74.44%	77.61%	73.82%
`idf`	77.20%	74.70%	77.74%	73.76%
`delta`	77.20%	74.71%	77.74%	73.76%
`rf`	77.16%	74.59%	77.21%	74.03%
`ig`	77.16%	74.49%	77.59%	73.79%

(d) Performance of the balanced label propagation classifier in the second round of experiments.

Scheme	SVM	Label propagation
`tf`	81.35%	77.06%
`chi^2`	81.78%	77.10%
`idf`	82.09%	79.03%
`delta`	81.94%	79.08%
`rf`	81.49%	77.16%
`ig`	81.56%	77.12%

(e) The results of the third set of experiments showing the balanced accuracies of the SVM and label propagation classifier on the entire data set.

classified the network nodes using the graph itself. Again, because of the imbalanced nature of the data set, we tested the label propagation classifier with the binary starting matrix Y (as first proposed in [25]) and with the starting matrix Y, adjusted to the class frequencies, as proposed in Sect. 3.2. In this set of experiments, we trained the classifiers on 90% (27.000 nodes) of the data set.

In all experiments we evaluated the accuracy of the classifiers using the *balanced accuracy* metric. This is the metric used in the PAKDD'15 Data Mining Competition and is defined as

$$\frac{\frac{|\{\text{Correctly classified male customers}\}|}{|\{\text{All male customers}\}|} + \frac{|\{\text{Correctly classified female customers}\}|}{|\{\text{All female customers}\}|}}{2}. \quad (3)$$

Experimental results. The first set of experiments, shown in Table 3a, shows that there is a large difference in the performance of different classifiers. Similarly to Grčar et al. [6], the simple centroid classifier performs well on feature vectors extracted from several different homogeneous networks. However, the classifier is still consistently outperformed by the SVM classifier if the instance weights of the classifiers are set according to the class sizes. We conclude that the optimal classifier for the methodology, introduced in [6], depends on the data set.

The results of the second set of experiments are shown in Tables 3b, 3c and 3d. When comparing the results of the two label propagation approaches the

results show that label propagation with adjusted starting matrix has large impact on the performance of the classifier, as the balanced accuracy increases by 1–2% in the case of the A- and D-level network and even more in the case of B- and C-level networks. Different heuristics used in construction of homogeneous networks also affect the final performance of all three classifiers. No heuristic consistently outperform the others, meaning that the choice of heuristic is application dependent. The last conclusion of the second round of experiments is that the computationally demanding propositionalization method does not outperform the label propagation method. In all four networks choosing a correct heuristic and correct weights for the starting matrix allows the label propagation method to perform comparably to the SVM classifier.

Table 3e shows the results of the third set of experiments. Here the propositionalization-based approach clearly outperforms the label propagation algorithm. It is possible that this effect occurs because the network propositionalization approach, in particular the SVM classifier, require more training examples (compared to the network propagation classifier) to perform well. The effect of using weighting heuristics in the construction of the homogeneous networks is also obvious. With both classification methods the adjusted `delta` and `idf` heuristics perform best.

6 Conclusions and further work

While network analysis is a well established research field, analysis of heterogeneous networks is much newer and less researched. Methods taking the heterogeneous nature of the networks into account show an improved performance [2]. Some methods like RankClus and others presented in [20] are capable of solving tasks that cannot be defined with homogeneous information networks (like clustering two disjoint sets of entities). Another important novelty is merging network analysis with the analysis of node data, either in the form of text documents or results obtained from various experiments [5, 8, 6].

The contributions of the paper are as follows. By setting the weights of the initial class matrix proportionally to the class value frequency, we improved the performance of the label propagation algorithm when applied to a highly imbalanced data set. We adapt heuristics, developed primarily for use in text mining, for the construction of homogeneous networks from heterogeneous networks. Our results show that the choice of heuristics impacts the performance of both label propagation classifier and classifiers based on the propositionalization approach of [6]. We also present a variation of the label propagation approach, described in [25].

In future we wish to detailly analyze the network-construction heuristics and their performance in classifiers applied to homogeneous networks. We plan to design efficient methods for propositionalization of large data sets and decrease the computational load of PageRank calculations by first detecting communities in a network. This "pre-processing" will allow us to shrink the network on which PageRank calculation are performed.

We plan to test the methods, described in this paper, on publicly available data sets such as the DBLP, Cora and CiteSeer databases. The presented heuristics shall be evaluated on the methodology for mining text enriched heterogeneous networks presented in [6]. For that one has to construct a heterogeneous network in which the central node represent genes, connected to the response of plants against an infection. We will enrich the nodes with papers from the PubMed database which mention the genes.

References

[1] Burt, R. and Minor, M. (1983). *Applied Network Analysis: a Methodological Introduction*. Sage Publications.

[2] Davis, D., Lichtenwalter, R., and Chawla, N. V. (2011). Multi-relational link prediction in heterogeneous information networks. In *Proceedings of the 2011 International Conference on Advances in Social Networks Analysis and Mining*, pages 281–288.

[3] Debole, F. and Sebastiani, F. (2004). Supervised term weighting for automated text categorization. In *Text mining and its applications*, pages 81–97. Springer.

[4] D'Orazio, V., Landis, S. T., Palmer, G., and Schrodt, P. (2014). Separating the wheat from the chaff: Applications of automated document classification using support vector machines. *Polytical Analysis*, 22(2):224–242.

[5] Dutkowski, J. and Ideker, T. (2011). Protein networks as logic functions in development and cancer. *PLoS Computational Biology*, 7(9).

[6] Grčar, M., Trdin, N., and Lavrač, N. (2013). A methodology for mining document-enriched heterogeneous information networks. *The Computer Journal*, 56(3):321–335.

[7] Han, E.-H. and Karypis, G. (2000). Centroid-based document classification: Analysis and experimental results. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 424–431. Springer.

[8] Hofree, M., Shen, J. P., Carter, H., Gross, A., and Ideker, T. (2013). Network-based stratification of tumor mutations. *Nature Methods*, 10(11):1108–1115.

[9] Jeh, G. and Widom, J. (2002). SimRank: A measure of structural-context similarity. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 538–543. ACM.

[10] Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21.

[11] Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

[12] Kondor, R. I. and Lafferty, J. D. (2002). Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the 19th International Conference on Machine Learning*, pages 315–322.

[13] Kralj, J. (2015). Mining heterogeneous citation networks. In *Proceedings of the 19th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 672–683.

[14] Kwok, J. T.-Y. (1998). Automated text categorization using support vector machine. In *Proceedings of the 5th International Conference on Neural Information Processing*, pages 347–351.

[15] Lan, M., Tan, C. L., Su, J., and Lu, Y. (2009). Supervised and traditional term weighting methods for automatic text categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):721–735.

[16] Manevitz, L. M. and Yousef, M. (2002). One-class SVMs for document classification. *Journal of Machine Learning Research*, 2:139–154.

[17] Martineau, J. and Finin, T. (2009). Delta TFIDF: An Improved Feature Space for Sentiment Analysis. In *Proceedings of the Third AAAI Internatonal Conference on Weblogs and Social Media*, San Jose, CA. AAAI Press.

[18] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

[19] Storn, R. and Price, K. (1997). Differential evolution; a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

[20] Sun, Y. and Han, J. (2012). *Mining Heterogeneous Information Networks: Principles and Methodologies*. Morgan & Claypool Publishers.

[21] Sun, Y., Yu, Y., and Han, J. (2009). Ranking-based clustering of heterogeneous information networks with star network schema. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 797–806.

[22] Tan, S. (2006). An effective refinement strategy for KNN text classifier. *Expert Syst. Appl.*, 30(2):290–298.

[23] Vanunu, O., Magger, O., Ruppin, E., Shlomi, T., and Sharan, R. (2010). Associating genes and protein complexes with disease via network propagation. *PLoS Computational Biology*, 6(1).

[24] Zachary, W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473.

[25] Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2004). Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16(16):321–328.

AIDA: an application of sequential pattern mining

Mirjana Mazuran, Matteo Simoni, and Letizia Tanca[1]

Politecnico di Milano – Dipartimento di Elettronica, Informazione e Bioingegneria
Piazza Leonardo da Vinci 32, 20133 Milano (Italy)
mirjana.mazuran@polimi.it, matteo.simoni@mail.polimi.it,
letizia.tanca@polimi.it

Abstract. An index is a data structure that improves the speed of data retrieval, thus facilitating query-answering. Data indexing is as fundamental as costly, since space and time are needed to construct and maintain the index data structure. In this paper we propose a new approach and tool for automatic index maintenance based on data mining. Given a log of operations and a set of user-specified queries, AIDA uses sequential pattern mining to learn to predict future occurrences of desired queries and maintain their indices on-demand.

1 Introduction

An index is a data structure that improves the speed of data retrieval facilitating query answering. Indices are built to support searching based on a search key, that is, in the relational database context, an attribute or a set of attributes: given a query imposing some conditions on an certain attribute, if the attribute has an associated index, finding the tuples satisfying the required constraints is much quicker. This improvement is fundamental but does not come for free, as space and time resources are needed to construct and maintain the index data structure. Changes to the data (such as insertions, updates, deletions, etc.) require the update of indices and maintenance costs depend on the kind of index. *Dense* indices have higher costs because an index record must exist for every search-key value; *sparse* indices have lower costs because each index record corresponds to a sequence of search-key values.

Indices are designed by keeping into account the queries the system deals with. Due to construction and maintenance costs, it is usually not possible to build an index for every single query, thus, depending on system requirements, some queries are considered more important (eg. because they are expensive) and the set of indices to adopt is based on the attributes they require. Every time a change in the data happens, all the involved indices must be updated. This continuous maintenance of indices is paramount to keep them up-to-date in particular in the case of queries that are frequently executed; however, for queries that are not so frequent, on-demand maintenance can be considered as an alternative, that is, the index must not be updated every time the data

change, but can be updated only before the related query is executed. Thus, if we are able to predict when an expensive, not very frequent query is going to be executed, we can perform index maintenance in time to use them.

In this paper we introduce a new technique for automatic indexing based on *Sequential Pattern Mining* (SPM), that creates a model of the execution of expensive queries and then adopts it to predict when the query is going to arrive in order to perform on-demand index maintenance. SPM [10] is a data-mining technique to find relevant patterns in data sequences. In Section 2 below, we review the state of the art of existing systems for automatic index management while in Section 3 we propose AIDA, our new method. In Section 4 we show some experimental results obtained with the AIDA tool and we conclude in Section 5 by stating our contributions and future developments.

2 State of the art

The capability to quickly answer user requests is fundamental in all systems that deal with data retrieval. Many techniques used to support query optimization consist in maintaining statistical information about the data; however, this information needs to be constantly and quickly kept up to date [1].

Index design and maintenance is a laborious process, thus, methods to support it have been investigated since the 70s [3] and many tools have been proposed [4–8] that usually take as input a log of the operations performed on data and produce the suggested indices. Most of them are either offline (periodically take the log and analyze it) or online (constantly monitor real-time operations performed on data) and can be manual, semi-automatic or automatic. Moreover, the majority of them is reactive, i.e., they update or suggest indices after a target expensive query has been executed, in order to have them ready for its next execution. A drawback is that the system can adapt (eg. suggest or update some indices) only after an undesired event has happened (eg. an expensive query has arrived). To overcome this limitation a proactive approach was introduced in [9] whose aim is to predict when a target query is going to be executed and update its indices before it arrives. Given a log of operations and a query, the method applies linear regression and other related techniques to learn to predict when the query is going to be executed again and to update its indices in time. We use a similar approach, but propose a different learning method: in [9], the prediction model of a query is based on linear regression and related techniques and it is based only on the timespans between different occurrences of the same query in the past; instead, our approach looks for *sequences of operations* whose last one is the target query. If some of these sequences recur in the log, then we expect them to recur also in the future and thus use them to predict future occurrences of the target query. A similar technique has also been investigated in the field of process mining [14].

To learn a model based on frequent sequences we use SPM [12], which is similar to frequent itemsets mining technique but considers sequences, i.e., the order of the items is important. Given a set of sequences and a frequency threshold,

the aim of SPM is to find frequent subsequences, that is, subsequences whose frequency is above the given threshold. Different approaches to SPM have been investigated and a lot of algorithms have been proposed [11]: we adopt PrefixSpan [13], that uses a pattern-growth approach where, starting from frequent patterns containing one element, it progressively increases the length of the found patterns in order to find the frequent ones.

3 The AIDA approach

AIDA [2] consists of two phases:

1. *Training*: given a log file containing past activities (that is, the sequence of operations, each one with a timestamp (Figure 1a), performed so far by the DBMS) and a user-specified target query q_B, AIDA learns a model for predicting future executions of q_B; 2)
2. *Using*: given the real-time sequence of operations being performed on the data, AIDA uses the constructed model to predict the arrival of query q_B and to perform index maintenance before the query is executed.

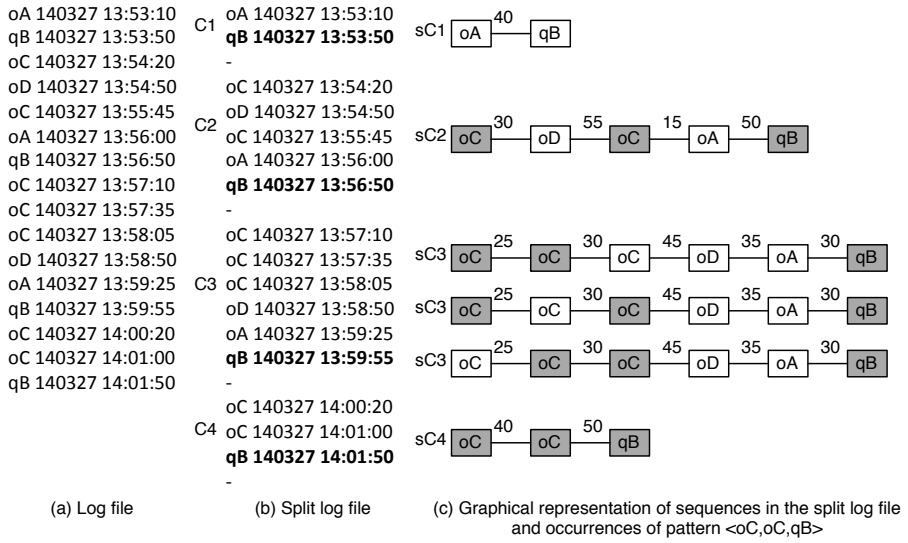


Fig. 1. Training AIDA

Background and notation. A sequence $\langle e_1, e_2, \ldots, e_I \rangle$ is an ordered list of events. Given two sequences $\alpha = \langle a_1, a_2, \ldots, a_n \rangle$ and $\beta = \langle b_1, b_2, \ldots, b_m \rangle$, α is a subsequence of β if there exist integers $1 \leq j_1 < j_2 < \ldots < j_n \leq m$ such that $a_1 = b_{j1}, a_2 = b_{j2}, \ldots, a_n = b_{jn}$. Consider a set of sequences $\mathcal{S}_e$, containing three sequences $s_1 = \langle a, b, c, d, c \rangle, s_2 = \langle a, d, c, b \rangle, s_3 = \langle f, a, b, d, c, b \rangle$; $\alpha = \langle a, b, c \rangle$ is a subsequence of s_1 and s_3 while it is not a subsequence of s_2. Given a set of

sequences $\mathcal{S}$ and a sequence α, the *absolute support* of α in $\mathcal{S}$ is the number of sequences in $\mathcal{S}$ such that α is their subsequence; while the *relative support* of α in $\mathcal{S}$ is the ratio between its absolute support and the total number of sequences in $\mathcal{S}$. The absolute support of α in $\mathcal{S}_e$ is 2 while its relative support is $2/4 = 0.5$.

3.1 Training AIDA

Given a log file containing query q_B and other operations o performed on the data (eg. insertions, deletions, updates, queries, etc.), AIDA learns a model to predict future executions of q_B. In particular:

- first, the log and a user-defined *minsupp* threshold are given as input to the PrefixSpan algorithm that outputs the set of frequent patterns;
- second, the found frequent patterns are further analyzed: (i) only those ending with q_B are kept and (ii) each pattern is enriched with temporal information to support more precise predictions.

If we break the log every time we find q_B (Figure 1b), we obtain several chunks, each one represents a history of events that brought, as the final event, to the execution of q_B (in Figure 1: $C1, C2, C3, C4$). By mining frequent patterns from them, we find the sequences of events that frequently led to q_B and use this succinct information to predict future executions of q_B.

Algorithm 1 `TrainingAIDA` (logfile,q_B,*minsupp*).

```
 1: chunks=splitLog(logfile,q_B)
 2: seqPatterns=PrefixSpan(chunks,minsupp)
 3: for all s ∈ seqPatterns do
 4:    if last event in s differs from q_B then
 5:       delete s from seqPatterns
 6:    end if
 7: end for
 8: for all s ∈ seqPatterns do
 9:    for all c ∈ chunks do
10:       if s is a subsequence of c then
11:          update timespans of s
12:       end if
13:    end for
14: end for
15: return seqPatterns
```

The training phase is formalized in Algorithm 1. After breaking the log sequence into subsequences terminating with q_B (line 1), AIDA inputs these sequences and a user-defined *minsupp* threshold to PrefixSpan, which outputs all the frequent subsequences (line 2). Given the log in Figure 1 and *minsupp*=0.5, some frequent patterns are: $\langle o_A, q_B \rangle$ (with support 3/4), $\langle o_C, o_D \rangle$ (support=1/2),

$\langle o_C, o_c, q_B \rangle$ (support=3/4). We keep only the patterns that end with q_B because they are the ones we can use for prediction purposes (lines 3-7 in Algorithm 1). In fact, pattern $\langle o_A, q_B \rangle$ gives us useful information about q_B because it tells us that frequently q_B happens after operation o_A has been executed. On the other hand the pattern $\langle o_C, o_D \rangle$ does not give us any useful information to predict the execution of q_B, thus we are not interested in keeping it.

Each pattern is enriched with one timespan for each pair of consecutive operations (lines 8-14 in Algorithm 1). This time information is necessary for the index maintenance phase; indeed, consider the pattern $\sigma = \langle o_C, o_C, q_B \rangle$: when the system detects an occurrence of o_C, it waits to see if another one arrives and, as soon as this is the case, starts updating the indices of q_B. However, this is not a reliable approach because the second occurrence of o_C might arrive well after the first one or not arrive at all. This is why, to give more accurate predictions, the search for frequent patterns is enriched by requiring that the timespan between one operation and the next one does not exceed a certain value. In this way we are able to express, for example, that if o_C occurs and, after a timespan of at most t_1, another occurrence of o_C arrives, then probably, after a timespan of at most t_2, the user-defined query q_B is going to be executed.

Given a frequent pattern p found by PrefixSpan, the time information to be associated with p is chosen by analyzing each chunk of the log file and, for each occurrence of p, taking the timespans between its events, as shown in Figure 1 for the pattern $\sigma = \langle o_C, o_C, q_B \rangle$. Note that some chunks do not contain any occurrence of σ ($C1$), some contain only one ($C2$ and $C4$) while others contain more than one ($C3$). The timespan assigned to each pair of consecutive operations is the average of the timespans in all p's occurrences. In the case of σ we compute t_1: the time between o_C and o_C, and t_2: the time between o_C and q_B. $t_1 = 47$ is the average of the timespans between o_C and o_C (85s in $C2$; 25s, 55s, 30s in $C3$ and 40s in $C4$) and similarly, $t_2 = 91.25$ is the average of the timespans between o_C and q_B (65s in $C2$; 140s, 110s, 110s in $C3$ and 50s in $C4$). To make our model more flexible, we add to each timespan a tolerance; different measures can be adopted, such as the variance, the standard deviation, the difference between the maximum and the minimum time divided by 2. However, the method can be easily extended to support different ad-hoc measures. At the end of this process, we have a model represented in terms of frequent patterns ending in the user-defined query and enriched with a temporal dimension (see an example in Figure 2).

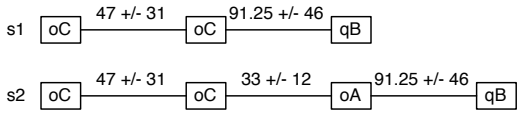


Fig. 2. Using AIDA

3.2 Using AIDA

The idea behind AIDA, formalized in Algorithm 2, is to check at real-time each operation requested to the DBMS and see if it is possibile to match it to an event in the sequence patterns; if it is possible we wait for another operation that matches the next event in the sequence. If such operation arrives within the timespan specified by the pattern we can match the corresponding event in the sequence and wait for the next one. If the desired operation does not arrive in time we continue considering the other patterns. Moreover, each time we match an event we decide whether it is time to schedule indices' maintenance; to do so we compute an estimate of the time left before the arrival of the user-defined query by summing all the times between the events we have not matched yet in the pattern. If the obtained time is greater than a user-specified *mintime* threshold we do not start updating the indices, otherwise we trigger the maintenance. Moreover, as the matching process goes on, if we reach the last event before the user-define query we start updating the indices regardless of the time estimation.

Algorithm 2 `UsingAIDA` (seqPatterns,q_B,t_{index}).

```
 1: partialSpList = seqPatterns
 2: for all o in run-time operations do
 3:    for all sp ∈ partialSpList do
 4:       if sp is time-valid then
 5:          if next node in sp is q_B then
 6:             maintain indices
 7:          else
 8:             compute estimated time before q_B
 9:             if estimated time <= t_index then
10:                maintain indices
11:             end if
12:             increment node in sp
13:          end if
14:       else
15:          remove sp from partialSpList
16:       end if
17:    end for
18: end for
```

Example 1. Consider pattern s_1 in Figure 2, let $mintime = 30s$ and suppose an occurrence of o_C arrives to the DBMS. AIDA matches it to the first event in s_1 and computes the estimate time left before the arrival of q_B as 91.25s - 46s = 45.25s (line 11 in Algorithm 2). Since the result is greater than *mintime* we do not start the update of the indices. Now, suppose that:

1. a second occurrence of o_C arrives after at least 47s - 31s but not later than 47s + 31s w.r.t. the first match. We match it to the second event in s_1. This event

is the last one in the sequence before the arrival of q_B: we expect to match q_B next, thus, we do not compute the estimate time left but immediately trigger the update of the indices (line 8 in Algorithm 2);

2. a second occurrence of o_C arrives after more than 47s + 31s w.r.t. the first match (or does not arrive at all). The pattern has expired, thus, we abandon the exam of that pattern (line 18 in Algorithm 2).

Example 2. Consider pattern s_2 in Figure 2, let $mintime = 50s$ and suppose an occurrence of o_C arrives. We match it to the first event in s_2 and we compute the estimate time left before the arrival of q_B as (33s - 12s) + (91.25s - 46s) = 66.25s. Since it is greater than *mintime* we do not start the update of the indices. Now, suppose another occurrence of o_C arrives in time (i.e. after at least 47s - 31s but not later than 47s + 31s w.r.t. the first one). We match the second event in s_2 and compute the estimate time of q_B as 91.25s - 46s = 45.25s. Since this time is lower than *mintime* we immediately start updating the indices. Note that we started the update of the indices before reaching the end of the pattern because the computed estimate time suggests that if we wait any longer there will not be enough time to update the indices to use them when q_B arrives.

Note that, every time an incoming operation is matched to the first node of a pattern, a new instance of that pattern is activated for future matchings and all active instances are evaluated independently. Given the patterns in Figure 2, an incoming o_C will be matched to the first node of each pattern, thus one instance of each pattern will be activated. If a second o_C arrives, it will be possibly matched to the second node of the two active pattern instances, however, it will also be matched to the first node of both patterns in Figure 2, thus, two new instances of these patterns will be activated.

4 Experimental results

We developed a Java prototype where, given a log file, the DB administrator specifies the set of queries the system should learn to predict. For each query the user has to specify a *minsupp* threshold needed to construct the model and a *mintime* threshold that is an estimate time for indices maintenance which is needed during the prediction phase.

We performed a study on the accuracy of AIDA's predictions using precision and recall. Let:

c_1 : the number of times AIDA predicts a query q (and thus updates its indices) and the query is actually executed;

c_2 : the number of times AIDA predicts a query q (and thus updates its indices) and the query is not actually executed;

c_3 : the number of times AIDA does not predict a query q (and thus does not update its indices) and the query is actually executed.

Then, precision is an indicator of the correctness, it indicates the number of correct predictions over the total number of predictions:

$$precision = \frac{c_1}{c_1 + c_2}$$

The recall is an indicator of completeness, it indicates the number of correct predictions over the total number of query executions:

$$confidence = \frac{c_1}{c_1 + c_3}$$

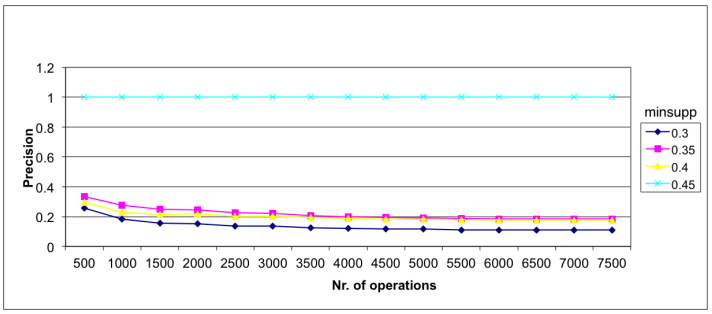


Fig. 3. Precision study

We used a dataset containing operations performed by users before an important event and the aim is to predict future occurrences of such event. In particular, in our experiments we use a log file containing 14451 operations, consider a single target query q and extract temporal sequential patterns considering different *minsupp* thresholds. As a result, the tool found 650 chunks containing an average of 10 events each. We perform the experiments on the log file using cross-validation (60% of data for training and 40% for testing). The models we find contain, depending on the *minsupp* threshold (0.3, 0.35, 0.4, 0.45), between 2 and 5 patterns with a length of 2 to 4 events. Lower support thresholds produce higher amounts of patterns which are usually more specific that is, longer in terms of the number of operations they involve; on the other hand, the higher the support threshold the lower the number of patterns whose length also tends to decrease.

After model construction, we analyze the operations coming to the DBMS and every 500 operations compute the precision and recall of the data analyzed so far. Figure 3 and Figure 4 show the results of our study. The first thing we notice from the results is a difference in the performances of the different models. In particular, a model based on the use of patterns with higher support has higher precision but lower recall while a model based on the use of patterns with lower support has lower precision but higher recall. To understand the meaning of this result we have to consider that patterns with high support are

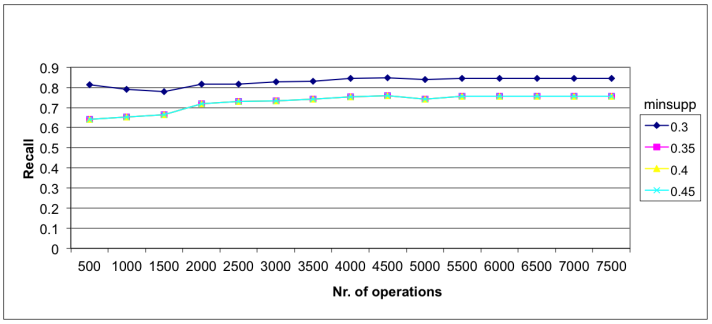


Fig. 4. Recall study

usually short, while patterns with low support are longer; therefore we can say that shorter sequences have a higher probability of being recognized thus they allow to obtain high precision while longer sequences have a lower probability of being recognized, thus have a lower precision.

5 Conclusion

Our new approach to automatic index maintenance based on the use of sequential pattern mining from the log of operations performed on data, to predict the arrival of a target query and start the update of its indices before its execution. Future works include the automatic identification of the queries to monitor, and of the most convenient support thresholds for each one. Moreover, we are currently expanding the experimental results by considering different SPM algorithms with the aim of comparing different performance costs and by keeping into account unexpected parameters such as drifts in the query distribution.

References

1. P. B. Gibbons, Y. Matias, V. Poosala. “Fast incremental maintenance of approximate histograms”. ACM Trans. Database Syst. 27(3): 261-298 (2002).
2. M. Mazuran, M. Simoni, L. Tanca, “AIDA: Automatic Indexing based on DAta mining”, SEBD 2015.
3. M. Hammer, A. Chan. “Index Selection in a Self-Adaptive Data Base Management System”. SIGMOD Conference 1976: 1-8.
4. C. Maier, D. Dash, I. Alagiannis, A. Ailamaki, T. Heinis. “PARINDA: an interactive physical designer for PostgreSQL”. EDBT 2010: 701-704.
5. K.-U. Sattler, I. Geist, E. Schallehn. “QUIET: Continuous Query-driven Index Tuning”. VLDB 2003: 1129-1132.
6. K. Schnaitter, S. Abiteboul, T. Milo, N. Polyzotis. “COLT: continuous on-line tuning”. SIGMOD Conference 2006: 793-795.
7. N. Bruno, S. Chaudhuri. “An Online Approach to Physical Design Tuning”. ICDE 2007: 826-835.
8. K. Schnaitter, N. Polyzotis. “Semi-Automatic Index Tuning: Keeping DBAs in the Loop”. PVLDB 5(5): 478-489 (2012).

9. A. Medeiros, A. Saraiva, G. Campos, P. Holanda, J.M. Monteiro, A. Brayner, S. Lifschitz. “Proactive Index Maintenance: Using Prediction Models for Improving Index Maintenance in Databases”. JIDM 3(3): 255-270 (2012).
10. R. Agrawal, R. Srikant. “Mining sequential patterns”. ICDE 1995: 3-14.
11. C. Mooney, J. F. Roddick. “Sequential pattern mining – approaches and algorithms”. ACM Comput. Surv. 45(2): 19 (2013).
12. T. Slimani, A. Lazzez. “Sequential Mining: Patterns and Algorithms Analysis”. CoRR abs/1311.0350 (2013).
13. J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, M. Hsu. “PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth”. ICDE 2001: 215-224.
14. M. Ceci, P. F. Lanotte, F. Fumarola, D. P. Cavallo, D. Malerba. “Completion Time and Next Activity Prediction of Processes Using Sequential Pattern Mining”. Discovery Science 2014: 49-61.

Multivariate Sequence Analysis with Transductive Classification

Zhao Xu and Koichi Funaya

NEC Laboratories Europe
Kurfürsten-Anlage 36, 69115 Heidelberg, Germany
{zhao.xu,koichi.funaya}@neclab.eu

Abstract. Multivariate sequence analysis is of growing interest for learning on data with numerous correlated time-stamped sequences. It is characterized by correlations among sequences and may not be separately analyzed as multiple univariate sequences. On the other hand, labeled data is usually expensive and difficult to obtain in many real-world applications. In this paper, we present a transductive learning framework for multivariate sequence classification. The graph-based transductive classification framework takes advantage of unlabeled sequences in addition to labeled data to enhance predictive performance. Additionally, we exploit PCA based method for graph construction to incorporate the correlation within the multivariate sequences. Experimental results on real-world data show that our approach achieves substantial improvements in classification performance.

1 Introduction

The growing popularity of social media, e-commerce and sensor systems has generated considerable interest in sequential data analysis, ranging from web site visit data mining to sensor data analysis in Internet of Things (IoT) systems. The sequential data in many applications involves numerous correlated observations received at each time point, such as data collected from multiple related sensors. An intrinsic property of multivariate sequential data is the correlations between different variables. So it may not be modeled as multiple independent univariate sequences.

In many real-world applications, there is usually a large amount of unlabeled data but limited labeled data, which can be difficult and time consuming to obtain. The aim of this work is to exploit additional information about the distribution of both labeled and unlabeled data to provide better performance of multivariate sequence classification. We propose a graph-based transductive sequence classification framework, which also incorporate the correlations between sequences. In our framework, a weighted graph of sequence is constructed based on the distances of both labeled and unlabeled data to capture the underlying structures. In particular, we exploit PCA (principal component analysis) based similarity measure to effectively encode the correlations within the multivariate sequences. The unlabeled sequences can be classified by label propagation over

the constructed graph with a harmonic Gaussian field based method [22]. We evaluate the transductive classification framework with real-world datasets. The proposed approach demonstrates superior predictive performance.

The rest of the paper is organized as follows. We discuss related work in Section 2. Section 3 introduces the proposed transductive multivariate sequence classification framework. The experimental results are provided in Section 4. Finally the paper is concluded in Section 5.

2 Related Work

To improve the predictive performance by leveraging unlabeled data, some semi-supervised learning methods for sequential data classification have been proposed in the recent literature. Wei et al. introduced a self-training method based on one-nearest-neighbor classifier[17]. The method starts by training a classifier with labeled data, by which the unlabeled data is classified. Then the most confident unlabeled time series with the estimated labels are added to the training data. The classifier is retrained and the procedure repeated. Self-training based methods are dependent on the employed classifier, and the classification mistake can reinforce itself [3]. The SUCCESS method proposed by Marussy et al. is based on constrained hierarchical clustering [10]. The method clusters the whole set of time series, including labeled and unlabeled ones, using single-linkage hierarchical agglomerative clustering. Then the top-level clusters are labeled by their corresponding seeds. Clustering based semi-supervised learning methods usually rely on whether clustering algorithms can match the true data distributions [21]. These work on semi-supervised sequence classification mainly concentrate on univariate sequential data, whereas multivariate sequential data is of growing importance in many applications.

3 The Transductive Sequence Classification Framework

In this section, we will describe the graph-based transductive learning framework for multivariate sequential classification. Transductive learning is originally analysed in [15], which can enhance supervised learning by extracting additional information from the unlabeled data [3]. Here we assume that there are a set of L labeled sequences, denoted as $\{(s_1, y_1), (s_2, y_2), \ldots, (s_L, y_L)\}$, and a set of U unlabeled sequences, denoted as $\{s_{L+1}, s_{L+2}, \ldots, s_{L+U}\}$. Each sequence in the data is multivariate, represented with a matrix $s_i \in R^{D \times M_i}$ whose columns are D-dimensional vectors for M_i time frames. The sequences are evenly spaced, and can be of variable length. The goal of the proposed framework is to fit a model for label prediction based on the entire dataset by exploiting underlying structure of the data. Within the transductive learning framework, each multivariate sequence (MVS) is viewed as a vertex $v \in V$ of a graph, and all the MVS are linked with each other using undirected edges $e \in E$, which are associated with weights W computed as a function of distance between the involved sequences.

The labels of $\{y_{L+1}, y_{L+2}, \ldots, y_{L+U}\}$ can then be estimated by exploring the structure of the graph $\mathcal{G} = (V, E, W)$ through label propagation.

To effectively derive the underlying structures of a set of multivariate sequences, we first construct a weighted graph $\mathcal{G}$ represented as an adjacency matrix W of size $(L+U) \times (L+U)$, where L and U denote the numbers of labeled and unlabeled sequences, respectively. Each entry $W_{i,j}$ represents weight of an undirected edge $e_{i,j}$ between sequences i and j, and is formulated as a function of their distance $d_{i,j}$, i.e.

$$W_{i,j} = f(d_{i,j}),\ d_{i,j} \geq 0.$$

The mathematical form of the function $f(\cdot)$ can be arbitrary. Inspired with stationary kernels, $W_{i,j}$ can be defined as, e.g.:

$$\text{Squared exponential:}\quad \exp(-d^2/2\ell^2) \tag{1}$$

$$\text{Rational quadratic:}\quad (1 + d^2/2\alpha\ell^2)^{-\alpha} \tag{2}$$

$$\gamma\text{-exponential:}\quad \exp(-(d/\ell)^\gamma),\ 0 < \gamma \leq 2 \tag{3}$$

The weight monotonically decreases with the distance. The smaller the distance, the larger the weight and the stronger the connection between two sequences. The function $f(\cdot)$ can model how the weights decay with the distances, and is more flexible than directly using the distances (or similarity) as the weights of the edges. In the experimental analysis of Sec. 4, we use γ-exponential function with $\gamma = 1$.

The construction of adjacency matrix W is based on the distance between MVS which can well capture the characteristics of MVS and the latent structure of the dataset. The distance measure for univariate sequence has been largely investigated in the literature [16, 8, 4, 11]. However these methods can not be simply extended to MVS due to correlations between dimensions in MVS. The MVS distance measures fall into two categories: dynamic time warping (DTW) based methods [12, 2, 1] and PCA/SVD based methods [6, 13, 20, 7, 18]. In this work, we exploit the Eros method, which explores correlations between different dimensions of MVS based on principal component analysis (PCA) and provide superior performance over DTW based methods [20]. The Eros method extends Frobenius matrix norm to measure similarity between two MVS using principal components extracted with PCA. In particular, MVS similarity is computed as a weighted sum of cosine similarity of eigenvectors between two MVS matrices s_i and s_j:

$$Eros(i,j) = \sum_{k=1}^{D} \omega_k |\cos\theta_k|, \tag{4}$$

where θ_k denotes the angle between the k'th principal components of s_i and s_j. The weight ω_k specifies how much variability of the set of MVS can be explained

by the k'th principal components:

$$\omega_k = \sum_{i}^{N} \lambda_{k,i} / \sum_{k}^{D} \sum_{i}^{N} \lambda_{k,i}, \tag{5}$$

where $\lambda_{k,i}$ is the k'th eigenvalue of the MVS s_i. As the Eros method provides similarity between two MVS, we define distance as $d_{i,j} = 1 - Eros(i,j)$ for graph computation. The proposed framework allows incorporating different MVS modeling techniques. Although we employ the Eros distance in this work, it is natural to generalize to other MVS modeling techniques, and leverage multiple MVS modeling methods by graph combination.

The adjacency matrix W formulates the constructed graph for the entire MVS data. The smaller the distance of a pair of MVS, the larger the corresponding entry $W_{i,j}$. It is natural to assume that the closer sequences tend to have similar class labels. The constructed graph with the adjacency matrix W fully specifies proximity and underlying structure of the data. Labels of sequences can propagate to unlabeled sequences according to their proximity on the graph. In this work, we use a harmonic Gaussian field based method [22] for label propagation.

Each MVS is associated with an auxiliary random variable $z_i \in \mathbb{R}$, which represents soft label of the MVS. The distribution of the set of random variables z_i's is modeled with Gaussian Markov random field. In particular, the state of z_i is only conditioned on the connected random variables, and follows a Gaussian distribution. As the connected vertices should have similar labels, the energy, i.e. sum of clique potentials of a Markov random field, can be defined as [22]:

$$E(\mathbf{z}) = \frac{1}{4} \sum_{i,j} W_{i,j} (z_i - z_j)^2. \tag{6}$$

The distribution of the Gaussian field is

$$p_\beta(\mathbf{z}) \propto \exp\left(-\beta E(\mathbf{z})\right), \tag{7}$$

where the parameter $\beta = 1/T$ is usually called inverse temperature of the field. The minimum energy of the field can be achieved at $\mathbf{z}^*$, which satisfies

$$\Delta \mathbf{z}^* = 0$$

due to harmonic property [22]. Δ denotes combinatorial graph Laplacian, which is an essential component in graph-based methods:

$$\Delta = D - W, \tag{8}$$

where D is diagonal degree matrix with $D_{i,i} = \sum_j W_{i,j}$. To characterize the properties of the data explicitly in terms of matrix operations, the harmonic function is defined as:

$$\begin{bmatrix} D_{\ell,\ell} - W_{\ell,\ell} & -W_{\ell,u} \\ -W_{u,\ell} & D_{u,u} - W_{u,u} \end{bmatrix} \begin{bmatrix} \mathbf{z}^*_\ell \\ \mathbf{z}^*_u \end{bmatrix} = \mathbf{0}, \tag{9}$$

where $\mathbf{z}_\ell^*$ denotes the observed labels, and $\mathbf{z}_u^*$ is the unknown ones to be predicted. The Laplacian of the entire data is split into four corresponding blocks for labeled and unlabeled MVS. By solving the function (9), we have:

$$\mathbf{z}_u^* = (D_{u,u} - W_{u,u})^{-1} W_{u,\ell} \mathbf{z}_\ell^*. \tag{10}$$

The class assignments of the unlabeled sequences can then be obtained by thresholding the soft labels $\mathbf{z}_u^*$.

Additionally, a graph kernel can be obtained using the Laplacian Δ. In particular, the distribution of the Gaussian random field is:

$$p_\beta(\mathbf{z}) \propto \exp\left(-\frac{\beta}{4}\sum_{i,j} W_{i,j}(z_i - z_j)^2\right) \tag{11}$$

$$= \exp\left(-\frac{\beta}{2}\mathbf{z}^T \Delta \mathbf{z}\right). \tag{12}$$

One can find that the term $(\beta\Delta)^{-1}$ defines a graph kernel (covariance matrix). Considering the singular matrix issue, we further regularize the Laplacian as $\Delta = \Delta + \mathbf{I}/\sigma^2$, where $\mathbf{I}$ denotes identity matrix. This corresponds to remove zero eigenvalues of the Laplacian by regularizing its spectrum [14]. Finally, the graph kernel is computed as:

$$k(\mathcal{G}) = \left[\beta\left(\Delta + \mathbf{I}/\sigma^2\right)\right]^{-1}. \tag{13}$$

The kernel specifies the correlations among MVS based on spectral graph theory. Due to the inverse of the Laplacian, the correlation here is not a local one, but depends on all values of the whole dataset. Intuitively, this kernel can better capture the latent structure of a set of MVS. We will provide visualization of the graph kernels in the experimental analysis section.

4 Experimental Analysis

The experimental analysis of the proposed framework is carried out on two real-world datasets, uWave and AUSLAN, to evaluate the performance of the framework on multivariate sequence classification problem.

The Australian Sign Language dataset (AUSLAN) is originally collected by [5]. The dataset includes 95 Auslan signs, for each of which 27 examples are captured from a native signer using high-quality position trackers with totally 22 sensors. The signs are of variable length. The average length of each sign is approximately 57 frames. Fig. 1 visualizes the signs: *answer*, *happy*, *cold* and *building*. The task of the experiments is to assign the 22-dimensional sequences into one of 95 signs (classes).

The uWave dataset [9] is about gesture recognition using data collected with 3-axis accelerometers. The dataset involves gestures of eight users. These samples are recorded with a 3-axis accelerometer and represented in the form of

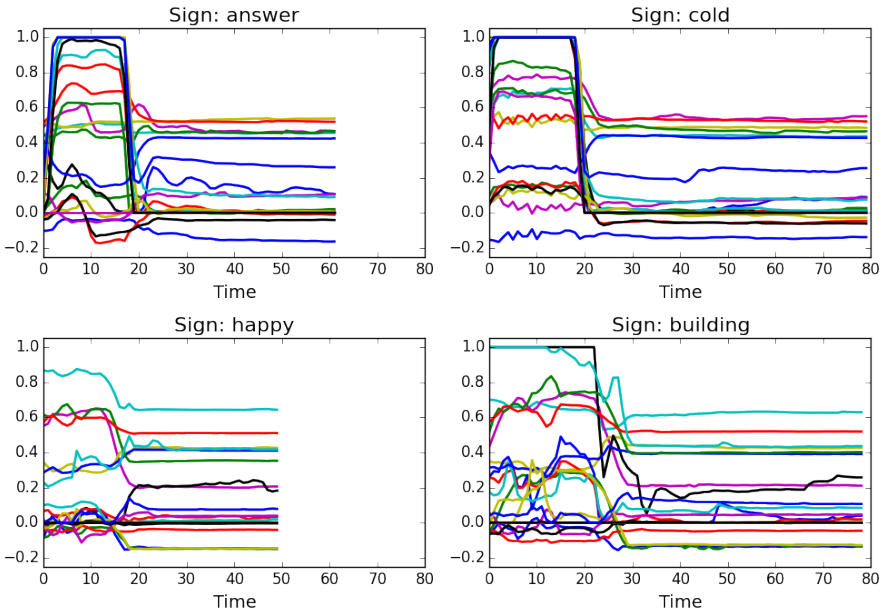


Fig. 1: Multivariate sequences of the Australian signs: answer, happy, cold and building.

Table 1: Misclassification ratio on the AUSLAN data

Labeled data	10%	20%	30%	40%	50%
Our approach	0.266	0.196	0.164	0.141	0.131
1NN	0.300	0.219	0.180	0.156	0.142

3-dimensional sequences. The length of the sequence are variable. In addition, the dataset is collected over multiple weeks to simulate the real situation. Users show high variations in a gesture over time. The task of the experiments is to perform user-dependent gesture recognition. Fig. 2 illustrates some gestures generated by two users at different days. One can see high variations in gesture sequences across users at different days.

The proposed framework is compared with the nearest neighbor classifier (denoted as 1NN), which is a commonly-used method in sequence classification problem with superior predictive performance [19]. In the experiments, we randomly select 10% (20%, 30%, ...) of data as labeled sequences and the rest as unlabeled ones for prediction. Every experiment is repeated 10 times, the averaged misclassification ratio is reported to measure the performance.

The experimental results on the AUSLAN data are summarized in Table 1 and Fig. 3. The experiments show that our approach substantially outperforms the baseline method, especially when the training data is few. We perform further

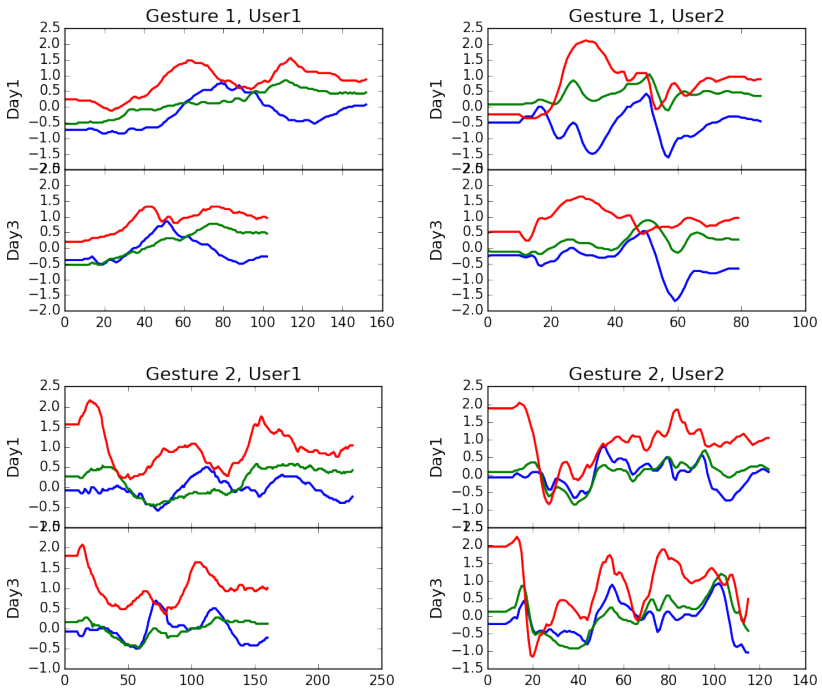


Fig. 2: Multivariate sequences of gestures generated by 2 users at different days.

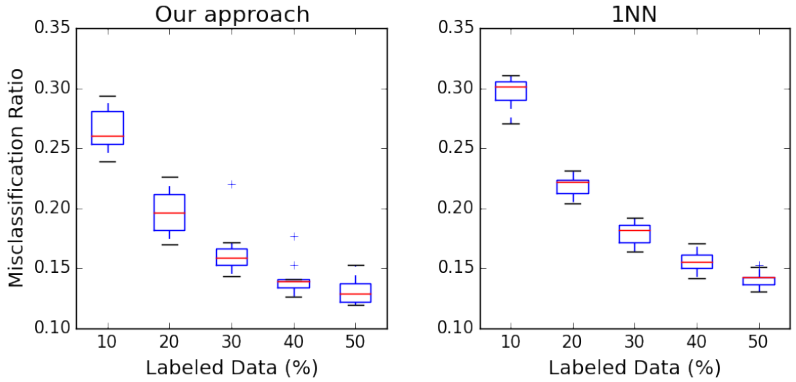


Fig. 3: Misclassification ratio on the AUSLAN data given 10% (20%, ..., 50%) of sequences with class labels. The box plots show variability across reruns.

analysis on our algorithm to better understand the results. Fig. 4 illustrates the structure and the corresponding graph kernel of the AUSLAN data with 100 example signs. One can find that the underlying structure of the data can be well

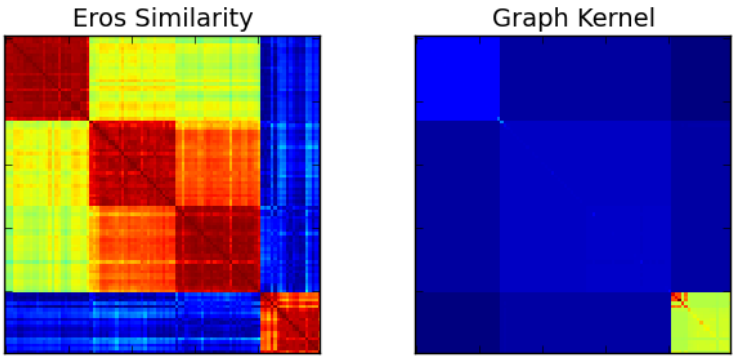


Fig. 4: The constructed graph and the corresponding graph kernel of the AUSLAN data.

captured by the adjacency matrix and formulated by graph kernel (right panel), which in turn results in improved performance. The experiments on the uWave data show similar tendency: the transductive classification framework provides superior predictive performance over the baseline method. Fig. 5 summarizes the experimental results. The experimental analysis demonstrates that exploiting the underlying structure of sequence data does improve the prediction performance.

5 Conclusion

We have presented a transductive learning framework to improve the performance of multivariate sequential classification. The framework exploits additional information on the distribution of both labeled and unlabeled sequential data with graph-based approach. Additionally the correlation within the multivariate sequences is also incorporated in graph construction. The proposed method achieves promising predictive performance in empirical analysis on real-world sequential datasets. An interesting avenue for future work would be to extend the framework to other types of time stamped sequential data, such as unevenly spaced sequence data.

References

1. Akl, A., Valaee, S.: Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, and compressive sensing. In: Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing (2010)
2. Banko, Z., Abonyi, J.: Correlation based dynamic time warping of multivariate time series. Expert Systems with Applications 39, 1281412823 (2012)
3. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-Supervised Learning. MIT (2006)

4. Esling, P., Agon, C.: Time-series data mining. ACM Computing Surveys 45(1), 1–34 (2012)
5. Kadous, M.W.: Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series. Ph.D. thesis, University of New South Wales (2002)
6. Krzanowski, W.J.: Between-groups comparison of principal components. Journal of the American Statistical Association 74, 703–707 (1979)
7. Li, C., Khan, L., Prabhakaran, B.: Real-time classification of variable length multi-attribute motions. Knowledge and Information Systems 10(2), 163183 (2005)
8. Liao, T.W.: Clustering of time series data-a survey. Pattern Recognition 38(11), 1857–1874 (2005)
9. Liu, J., Wang, Z., Zhong, L., Wickramasuriya, J., Vasudevan, V.: uwave: Accelerometer-based personalized gesture recognition and its applications. In: Proc. IEEE Int. Conf. Pervasive Computing and Communication (PerCom) (2009)
10. Marussy, K., Buza, K.: SUCCESS: a new approach for semi-supervised classification of time-series. ICAISC, LNCS 7894, 437–447 (2013)
11. Montero, P., Vilar, J.A.: Tsclust: An r package for time series clustering. Journal of Statistical Software 62(1), 1–43 (2014)
12. Rath, T.M., R, R.M.: Lower-bounding of dynamic time warping distances for multivariate time series. Tech. Rep. MM-40, University of Massachusetts (2002)
13. Shahabi, C., Yan, D.: Real-time pattern isolation and recognition over immersive sensor data streams. In: Proceedings of the 9th International Conference on Multi-Media Modeling (2003)
14. Smola, A.J., Kondor, R.: Kernels and regularization on graphs. In: Proceedings of the conference on Learning Theory (2003)
15. Vapnik, V., Chervonenkis, A.: On the uniform convergence of relative frequencies of events to their probabilities. Theory of Probability and Its Applications 16(2), 264–280 (1971)
16. Wang, X., Mueen, A., Ding, H., Trajcevski, G., Scheuermann, P., Keogh, E.: Experimental comparison of representation methods and distance measures for time series data. Data Mining and Knowledge Discovery 26(2), 275–309 (2012)
17. Wei, L., Keogh, F.J.: Semi-supervised time series classification. In: Proceedings of KDD. pp. 748–753 (2006)
18. Weng, X., Shen, J.: Classification of multivariate time series using locality preserving projection. Knowledge-based Systems 21(7), 581–587 (2008)
19. Xing, Z., Pei, J., Keogh, E.: A brief survey on sequence classification. SIGKDD Explorations 12(1) (2010)
20. Yang, K., Shahabi, C.: A pca-based similarity measure for multivariate time series. In: Proceedings of the 2nd ACM International Workshop on Multimedia Databases. pp. 65–74 (2004)
21. Zhu, X.: Semi-supervised learning literature survey. Tech. Rep. TR 1530, University of Wisconsin Madison, Department of Computer Sciences (2008)
22. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using Gaussian fields and harmonic functions. In: Proceedings of the 20th International Conference on Machine Learning (2003)

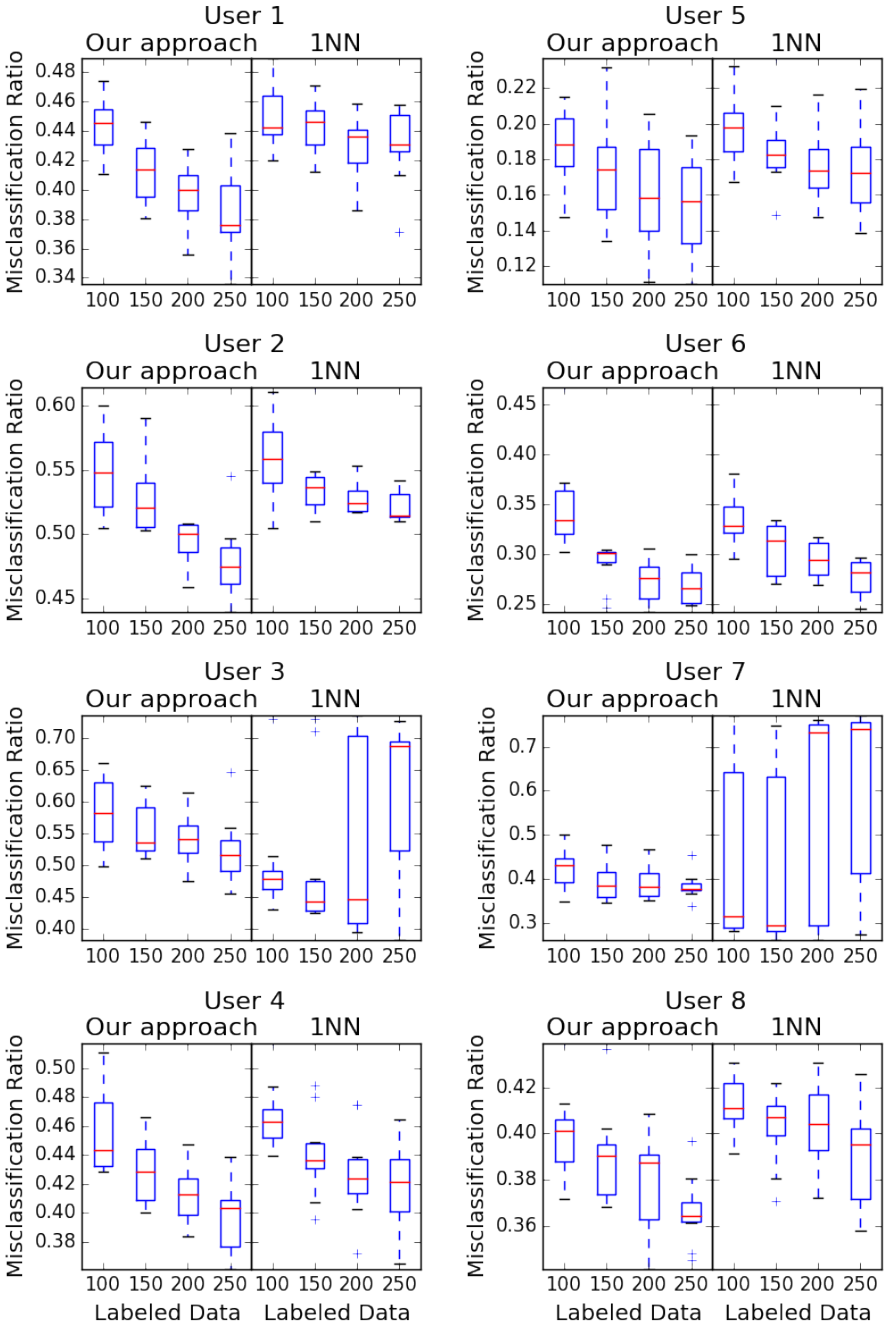


Fig. 5: Misclassification ratio on the uWave data given 100 (150, ..., 250) labeled sequences. The box plots show variability across reruns.

Evaluating a simple string representation for intra-day foreign exchange prediction

Simon Cousins and Blaž Žličar

Department of Computer Science,
University College London,UK
simon.cousins@cs.ucl.ac.uk
b.zlicar@cs.ucl.ac.uk

Abstract. This paper presents a simple string representation for hourly foreign exchange data and evaluates the performance of a trading strategy derived from it. We make use of a natural discretisation of the time-series based on arbitrary partitioning of the real valued hourly returns to create an alphabet and combine these individual characters to construct a string. The trading decision for each string is learnt in an incremental manner and is thus subject to temporal fluctuations. This naive representation and strategy is compared to the support vector machine, a popular machine learning algorithm for financial time series prediction, that is able to make use of the continuous form of past prices and complex kernel representations. Our extensive experiments show that the simple string representation is capable of outperforming these more exotic approaches, whilst supporting the idea that when it comes to working in high noise environments often the simplest approach is the most effective.

Keywords: Financial time series prediction, Discretisation, Parzen window estimators, Text kernels, Support vector machines

1 Introduction

The returns of financial time series (FTS) are renowned for being extremely noisy making it difficult to make predictions based upon these observations. This has led many previous authors to seek out novel and exotic representations of the time series that remove this noise and allow for better predictions to be made. In this paper we present an alternative view of the problem and introduce a simple discretisation operator on the observed returns. This discretisation, or binning, operator maps individual hourly returns from their original continuous form to a letter from an arbitrary, predefined alphabet. The letters corresponding to past returns are combined together to form a string representative of past price movements and we use this string to guide our trading decisions.

In this paper we examine 8 years of hourly data taken from four of the most actively traded currency pairs; AUD/USD, CHF/USD, EUR/USD and GBP/USD. The performance of our simple strategy is compared against support vector machines (SVMs), a popular machine learning algorithm for FTS

prediction. Extensive experiments are conducted using the SVM framework and a wide range of parameters are evaluated. The strategy that we present uses an incremental learning algorithm that records the average return of each unique market state representation (string). The decision on whether to trade is controlled by using an expected return threshold and confidence proxy, measured in terms of the number of times we have visited that state. The results indicate that our relatively simple approach is capable of outperforming its more complicated counterpart, supporting our claim that the simplest solution is likely to fare best when in noisy environments.

2 Previous work

This paper addresses the problem of FTS prediction from two aspects: (a) in terms of the complexity of an algorithm and (b) in terms of data representation techniques.

A large part of FTS research focuses on pattern discovery within the noisy data. The usual approach is to use a type of autoregressive model so as to predict the future price or return trajectories. While this approach has proven quite useful for the prediction of mean-reverting and persistent processes such as volatility, it is less successful in predicting the value or even directional movement of a price time-series. Machine learning has demonstrated some encouraging results for the prediction of FTS, with SVMs being one of the most popular approaches. One of the first applications of SVMs to FTS was presented in [1] and later extended in [2]. Both papers provide an empirical analysis of SVM-based FTS prediction and compare its performance against a number of other techniques including multi-layer back-propagation neural network and case-based reasoning. The experimental results in both paper suggest a superiority of SVM methods when compared to similar techniques, however they do outline the challenges of the SVM approach in terms of generalisation. In [3] the authors compare the performance of Least Squares SVMs (LS-SVMs)[1] to that of several autoregressive models as well as non-parametric models for both return and volatility prediction. They report a superior performance of the LS-SVM in terms of achieving higher directional prediction and better overall performance compared to that of other models used in their investigation. Further applications of SVMs for financial forecasting can be found in [4,5,6,7,8].

In terms of data representation, the prevalent approach is to work with continuous time series data, favouring returns instead of prices due to a number of statistical properties that they possess. Relatively less attention has been given to the study of the discretisation of financial time series, although there have been attempts to reduce the noise of a time series by using various quantisation techniques. Another way is to employ rule-based prediction methods that allow the incorporation of prior knowledge into the decision-making process. According to [9] rule-based forecasting involves two sources of knowledge (a) forecasting

[1] An extension of the original SVM that penalises the slack variables according to their squared value.

expertise (e.g. quantitative extrapolations and modelling) and (b) domain knowledge (practical knowledge about causal relations within particular field). Perhaps the most popular example of the latter in finance are methods with rules based on technical indicators[2]. These allow a researcher to include their expert knowledge into the forecasting process in the form of various thresholds and patterns applied to technical indicators that ultimately lead to discrete evaluation of the market at a specific point in time. Understandably, rule-based forecasting goes beyond rules based on technical indicators, e.g. assigning different considerations to level and trend of a time series, combining predictions of a number of models, separating models according to their forecast horizons etc. ([9]).

In this sense, machine learning is less restrictive compared to classical time series analysis since it does not necessarily demand that the process satisfies a specific set of assumptions. While classical econometrics normally steers clear of including technical indicators and other heuristics, machine learning techniques allow for an easy inclusion of such indicators without violating any statistical assumptions. In fact it is quite popular to use these technical indicators as building blocks of the feature space when predicting FTS with machine learning algorithms. Moreover, it extends the possibilities of data representation through the use of kernel functions. The majority of implementations use the Gaussian kernel function due to its rich representation ability. Also frequently used are Polynomial and Laplacian kernels. To the best of our knowledge, text-based kernels have not yet been applied for pattern recognition in FTS, except in the context of news analytics. For more details on kernel methods see for example [10].

3 Simple Strategy

In this section we discuss a simple representation for FTS and explain how to use it to construct a trading strategy. At each hourly interval we have prices for the open, high, low and close over that interval, given by O_t, H_t, L_t and C_t respectively, with each price being a positive real number. Our goal is to predict at the beginning of the price interval whether the price will increase or decrease over the interval, therefore our target is $y_t = \text{sign}\,(C_t - O_t) \in \{-1, +1\}$. To do this we look at the relative price movements, or returns, $r_t = (C_t - O_t)/O_t$ that have occurred leading up to time t. Rather than using the continuous values of the returns we form a partitioning of the real line $\mathbb{R}$ and label each sub-interval with a unique identifier i.e. a letter σ from alphabet Σ.

The approach we take is to use sub-intervals $(b_k, b_{k+1}]$, where $b_k < b_{k+1}$, that have roughly an equal number of members i.e. in the training sample there a roughly the same number of returns r_t corresponding to each sub-interval. This can be achieved by sorting the returns r_t and taking equally separated percentile points as the limits of the sub-intervals. An important point to consider is the

[2] Technical indicators would best be described as rule-based evaluations of the underlying time series where their mathematical formulae is not based on statistical theory but on an expert's domain knowledge instead.

sub-interval that contains both positive and negative values. Intuitively it makes sense to split this sub-interval into two separate sub-intervals either side of zero.

The mapping $A : \mathbb{R} \to \Sigma$ between returns and the alphabet is given by

$$A(r) = \begin{cases} \sigma_1, & b_0 < r \leq b_1 \\ \sigma_2, & b_1 < r \leq b_2 \\ \quad \vdots & \\ \sigma_{|\Sigma|}, & b_{|\Sigma|-1} < r \leq b_{|\Sigma|}, \end{cases} \tag{1}$$

where $b_0 = -\infty$ and $b_{|\Sigma|} = +\infty$. Using this notion each observation belongs to a given sub-interval, which we identify with a letter from our alphabet. The representation can be extended to include the past observations by concatenating the letters corresponding to past returns (see Figure 1).

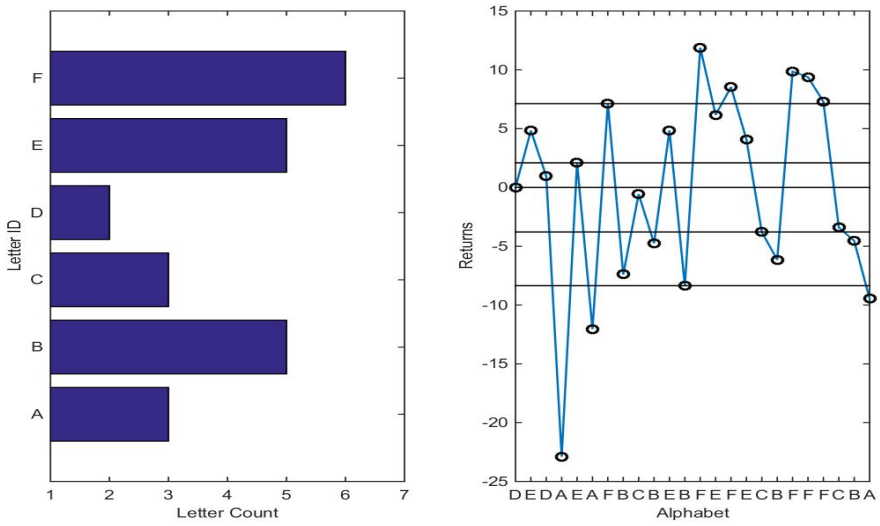


Fig. 1. Alphabet representation: left side displays letters (i.e. partitions) together with the number of examples (returns) that fall into individual partitions. Right side displays the time series of these same hourly returns over the period of 24 hours with each of the hourly returns being assigned a letter depending on which partition they fall into.

More formally, an alphabet Σ of $|\Sigma|$ letters is constructed by an arbitrary partitioning of the real line $\mathbb{R}$. Each return $r_t \in \mathbb{R}$ is mapped to a letter $\sigma_t \in \Sigma$. A string is constructed according to $s_t = \sigma_{t-1} \dots \sigma_{t-K}$, where K defines how many past returns we look at. Our goal is to come up with a prediction rule $g : \Sigma^K \to \{-1, 1\}$ indicating whether we believe the price will increase or decrease over the next interval. The feature space ϕ is constructed by mapping each

unique string $s \in \Sigma^K$ to binary vector with one non-zero entry corresponding to that particular string such that $\langle \phi(s), \phi(s') \rangle = 1$ if and only if $s = s'$, and zero otherwise. This is equivalent to a bag-of-words kernel where we only have a single word in each document.

We learn our predictor in an incremental manner by maintaining an individual weight w_s and count c_s for each string. This weight w_s is given by the sum of observed outcomes for that string i.e. the weight corresponding to string s at time T is given by $w_s = \sum_{t=1}^{T} y_t I[s_t = s]$, where $I[a]$ is the indicator function returning 1 if the predicate a is true and 0 otherwise. The count c_s simply measures the number of times we have seen string s, i.e. $c_s = \sum_{t=1}^{T} I[s_t = s]$.

We can think of our predictor as an incremental version of the Parzen window classifier, which is traditionally used in conjunction with feature map ϕ and kernel function $k(x, x') = \langle \phi(x), \phi(x') \rangle$. The Parzen window prediction function g_{pw} is given by

$$g_{\text{pw}}(x) = \text{sign}\left(\sum_{t=1}^{T} y_t k(x_t, x) \right),$$

which takes into consideration each training example. This is often referred to as the Watson-Nadaraya estimator, which is an estimate of the conditional probability of a class. Note that we no longer have to maintain previous examples as they can be captured by the primal representation of the predictor and we only have to maintain $|\Sigma|^K$ individual weights. In our experiments this remains a feasible primal representation as the maximum alphabet length and string length are both set to 5 meaning that $|\Sigma|^K \leq 3125$. In Algorithm 1 we present the simple string based trading algorithm and have introduced two additional variables, a threshold τ and minimum observation number ν. The threshold τ can be interpreted as the excess in probability of a given class occurring that is required to invoke a trading decision. The minimum observation number ν is used to ensure that we can have gathered enough information in order to make a decision. Together these variables control the level of confidence that we have in our trading decision. The dimension of our primal weight vector $\mathbf{w}$ depends on the size of the alphabet Σ and the number of past returns K that we examine i.e. $|\mathbf{w}| = |\Sigma|^K$. At each time step t we only update a single entry of $\mathbf{w}$, the one corresponding to the particular string observation s_t at that time i.e. w_{s_t}. We construct and update the counts $\mathbf{c}$ in a similar manner.

4 String subsequences strategy

In this section we examine an extension of the simple string strategy by measuring the impact that subsequences of strings, known as n-grams (NG), have on the probability of class membership i.e. given string s do we expect price to go up or down. Here we present a quick overview of the n-grams approach, a method for text representation popular especially in the fields such as computational linguistics [11] and bioinformatics [12]. One way a document can be

Algorithm 1 Algorithm to run simple strings trading strategy

SimpleStrings$(S, R, \Sigma, K, \tau, \nu)$

Require: S-string representation, R returns, Σ alphabet, K time steps considered, τ trade threshold, ν minimum observation number

Initialise weights $w_s = 0$ and counts $c_s = 0$ for each $s \in \Sigma^K$, Profit $R = 0$
for $t = 1 : T$ **do**
  Observe s_t, c_{s_t} compute $f(s_t) = \langle \mathbf{w}, \phi(s_t) \rangle = w_{s_t}$ and $\delta = w_{s_t}/c_{s_t}$
  if $\delta > \tau \cap c_{s_t} > \nu$ **then**
    Take long position, $p = 1$
  else if $\delta < -\tau \cap c_{s_t} > \nu$ **then**
    Take short position, $p = -1$
  else
    Do not trade, $p = 0$
  end if
  Observe return r_t, $y_t = sign(r_t)$
  Update profit $R \leftarrow R + pr_t$
  Update observation counts $c_{s_t} \leftarrow c_{s_t} + 1$
  Update weight vectors $w_{s_t} \leftarrow y_t + w_{s_t}$
end for

represented is in terms of substrings where each substring represents a feature of the underlying document. As its name suggests n-grams refers to n-number of adjacent characters in the alphabet with each n-gram type representing a type of substring (i.e. a feature). We can write such a mapping of a document d_l into a vector space characterised by n-grams as $\phi : d_l \to \phi(d_l) \in F \subseteq \mathbb{R}^{|\Sigma|^n}$.

As an example let us consider that the entire document is composed of the word "excellent", ($d_l =$ `excellent`) and we are interested in a 3-grams feature representation. The word "excellent" contains 7 unique 3-grams,

$$\texttt{excellent} \to [\ \texttt{exc xce cel ell lle len ent}\ ],$$

which will correspond to 7 non-zero entries in the feature mapping ϕ. This simple example shows that the dimensionality in real life problems can increase very quickly. One of the challenges associated with n-grams is the choice of the value of n. In practice this value is normally relatively low as the dimensionality problem basically does not allow for very high values of n. In terms of our strings subsequence strategy, note that the size of the feature space is now $|\Sigma|^k$, where $k \leq K$ is the size of the subsequences that we examine. However our feature space is no longer a binary vector with one non-zero entry, instead there is a non-zero entry for each subsequence that is present in the string. The approach used in Algorithm 1 can be simply adjusted to account for the use of n-grams. The observation $s = (s_1^p \dots s_k^p)$ is now decomposed to $k = \max(K - p + 1, 0)$ n-grams of length p, which results in $f(s) = \frac{1}{k} \sum_{i=1}^{k} w_{s_i^p}$. The weight vector and count updates occur in a synonymous manner to before, where we take each subsequence s_i^p in turn, updating its count $c_{s_i^p}$ and weight $w_{s_i^p}$. To keep our expressions as clear as possible we now drop the superscripts on s_i^p, when

the context is clear that we use a fixed length n-gram. The confidence of the trading decisions are controlled by the total count $c = \frac{1}{k}\sum_i c_{s_i}$ and the value $\delta = f(s)/c$. The value δ can once again be interpreted as an estimate that the difference between the probability that price will increase versus decrease over the next time period, given that we have observed string s and all past strings.

4.1 Time decay n-grams

The n-gram feature space that we have described thus far corresponds to the traditional one used in machine learning literature, however this has not been designed to take into consideration any temporal influences that may exist. For example, we would expect that more recent subsequences will have a stronger influence on the likely outcomes and should therefore have a greater weight placed upon them. To factor this into our representation we can introduce a simple decay function that weights subsequences according to their position in the string,

$$f(s) = \sum_{i=1}^{k} q_i w_{s_i} \text{ where } \sum_{i=1}^{k} q_i = 1 \text{ and } q_1 \geq q_2 \geq \cdots \geq q_k,$$

note that in traditional n-grams we effectively have $q_i = \frac{1}{k}$ for each k. We have to make a slight adjustment to the updates, which are now given by $c_{s_i} \leftarrow q_i + c_{s_i}$ and $w_{s_i} \leftarrow q_i y + w_{s_i}$, where $\mathbf{w} = (w_s)_{s\in\Sigma^k}$ maintains a weighted sum of the directional movements associated with each subsequence. We now show that the prediction rule g is equivalent to that of a Parzen window classifier constructed using this new time decay feature mapping.

Proposition 1. *Let $\phi(s)$ be the feature mapping associated with the time-decay n-gram kernel of length k given by*

$$k(s, s') = \sum_{u\in\Sigma^k} \sum_{i=1}^{k} \sum_{j=1}^{k} q_i q_j I[s_i = u] I[s_j = u],$$

where $\sum_{i=1}^{k} q_i = 1$ and $q_i \geq q_j$ if $i \geq j$. At time T the value of each component of the primal weight vector $(w_{s_i})_{s_i\in\Sigma^k}$ is given by $w_{s_i} = \sum_{t=1}^{T} y_t \sum_{i=1}^{k} q_i I[s_{t,i} = s_i]$, where $s_{t,i}$ corresponds to the i-th n-gram at time t. At time T the prediction function $g(s) = sign(\langle \mathbf{w}, \phi(s)\rangle) = sign\left(\sum_{i=1}^{k} q_i w_{s_i}\right)$ is equivalent to the Parzen window classifier given by $g_{pa}(s) = sign\left(\sum_{t=1}^{T} y_t k(s, s_t)\right)$.

Proof. By expressing the sum over kernel functions in terms of the subsequences present in string s we have

$$\sum_{t=1}^{T} y_t k(s, s_t) = \sum_{t=1}^{T} y_t \sum_{i=1}^{k} q_i \sum_{j=1}^{k} q_j I[s_{t,j} = s_i] = \sum_{i=1}^{k} q_i \sum_{t=1}^{T} y_t \sum_{j=1}^{k} q_j I[s_{t,i} = s_i],$$

which is equivalent to $\sum_{i=1}^{k} q_i w_{s_i}$. Therefore we see that both the Parzen window classifier and our simple average strategy will return the same prediction.

5 Experiments

We evaluate the performance of our proposed string based predictors by using hourly data taken from four of the most actively traded currency pairs; AUD/USD, CHF/USD, EUR/USD and GBP/USD. The extensive dataset consists of 50,000 observations for each currency, covering dates ranging from February 2005 to January 2013. Digressing briefly, the majority of previous experiments using machine learning for FTS prediction have focused on predicting stock returns and often report abnormal returns. We have chosen to focus on currencies due to their permanence and relatively stable prices, rather than the survivorship bias and tendency for an upward drift in prices that exists with stocks taken from indices such as S&P500 or the FTSE100. We investigate the performance from two perspectives: (a) the type of data used for decision making and (continuous vs. discrete i.e. alphabet) and (b) the complexity of the learning algorithm (Simple Strings vs. SVM).

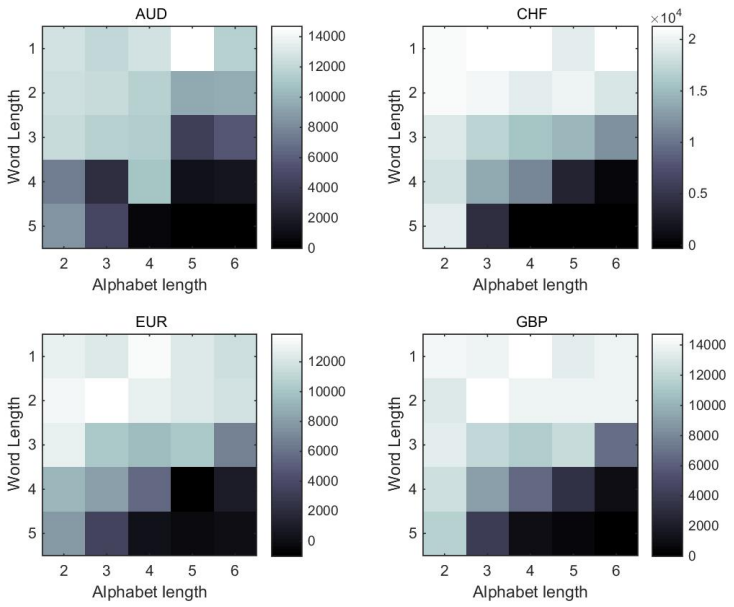


Fig. 2. Threshold $\tau = 0.00$: heat map shows a cumulative absolute return for the simple strategy for different alphabet and word lengths. The lighter the field the higher the absolute returns for a particular combination of alphabet and word lengths.

5.1 Simple Strings Strategy

Initially we run a simple strategy for a range of alphabet lengths $|\Sigma| = [2 : 6]$ and word lengths $K = [1 : 5]$. We set the minimum number of previously seen observations fixed at $\nu = 50$. The experiments are then conducted for a range of threshold values $\tau = 0.00$ and $\tau = 0.05$. Figures 2 and 3 display the cumulative return for $\tau = 0.00$ and $\tau = 0.05$, respectively. The heat maps reflect the cumulative absolute return achieved by the simple strings strategy for various values of alphabet and word lengths. We see that the best performance is achieved by strategies with short word and alphabet lengths. This resembles the empirical fact that there is little to no autocorrelation between returns and that the short look-back periods offer more information relative to longer periods.The performance does not seem to be overly sensitive to the size of the threshold τ, although the returns do seem to be slightly higher when $\tau = 0.05$ is used to assist the decision-making process.

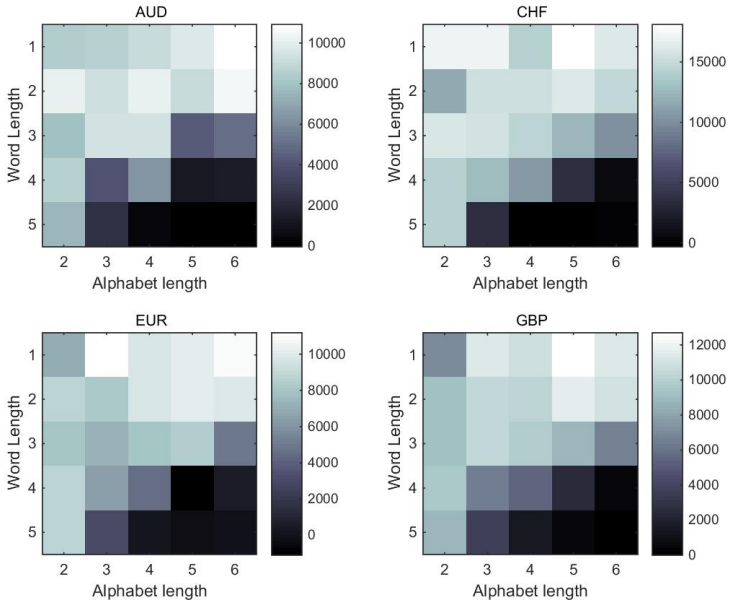


Fig. 3. Threshold $\tau = 0.05$: simple strategy applied with a non-zero trading threshold. Again, shorter alphabet and word lengths result in higher cumulative absolute returns.

5.2 SVM Strategy

To compare the performance of the simple string representation we trained a linear SVM classifier using the continuous representation of the respective string

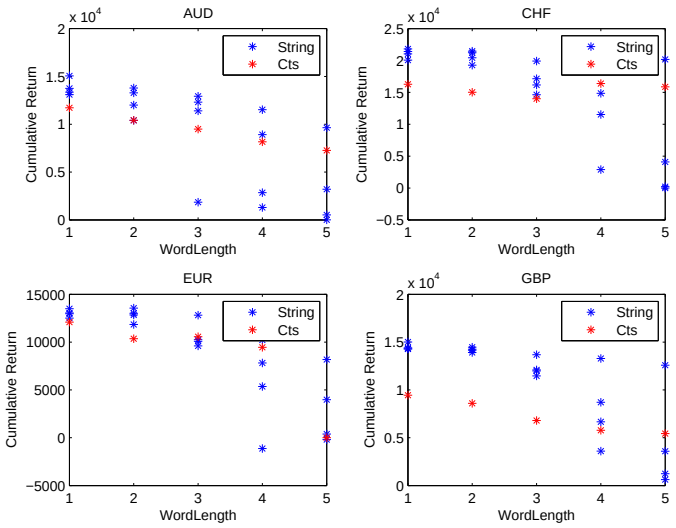


Fig. 4. Performance (cumulative return) comparison between linear SVM and simple string representation. Blue stars correspond to performance of different alphabet and word length string combinations, red stars are the linear SVM performance for continuous inputs of given length.

length i.e. $K = 5$ means that we used the last 5 returns to construct the input space for the SVM. Given the size of our dataset, we opted for a rolling window classifier approach, training for 1500 points, using a validation set of 500 points to choose the best regularisation parameter, and then testing the predictor on the subsequent 500 points. We compare the performance of the linear SVM with our string representation in Figure 4. We can see that in general the simple string representation outperforms a linear SVM trained using the continuous features. For most currencies and string parameterisations, the cumulative performance is higher than that of the continuous linear SVM. This results show that the simple discretisation and optimisation scheme can perform on par with more sophisticated learning algorithms. One explanation could be the fact the the strength of the SVM comes from the generalisation guarantees offered by the margin that it obtains. However in particularly noisy situations, like FTS, the margin that the algorithm obtained may not provide reasonable generalisation guarantees. Furthermore the solution of the SVM is constructed based upon those examples that lie at the margin or on the wrong side of it, therefore we see that solutions in noisy situations are comprised of examples that lie on the periphery of the underlying distribution whereas the approach we have taken simply uses an efficient representation of the class conditional mean of these samples.

5.3 String subsequences

To evaluate the performance of the string subsequences representation outlined earlier, we repeated the same experiments used for the simple strings, whilst varying the size of the n-grams and the weighting function q_i. For all of the experiments presented we used a simple decay factor of 10, meaning that $q_1/q_K = 10$. In Figure 5 we compare the performance of the string subsequence approach to our previous approach. We can see from the results that the n-gram implementation performs similarly to the original string approach, however what we can say is that it appears to be less sensitive to parameterisation. This is evident from a smaller amount of variance in the returns across a range of parameters. This shows promise for this approach for representation and something that could be studied more extensively in the future.

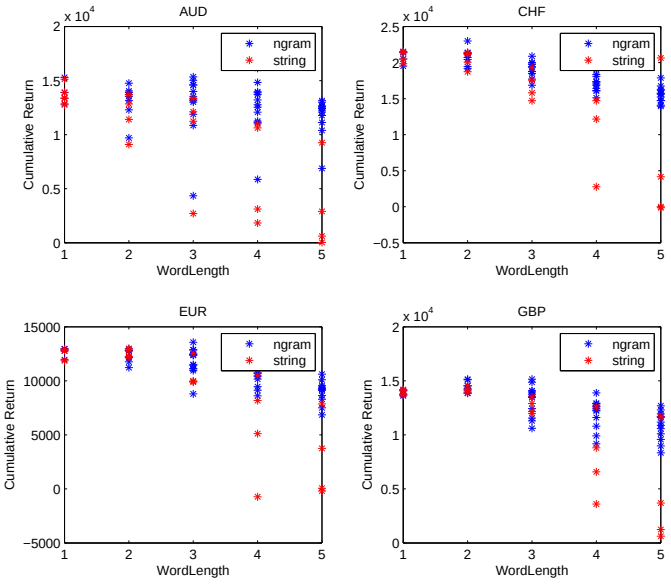


Fig. 5. Performance (cumulative return) comparison between n-gram and string representation. Blue stars correspond to performance of different n-gram combinations of n-gram length and alphabet length, red stars are the string performance for different alphabet lengths.

6 Conclusions

This paper presents a novel approach to FTS forecasting based on a simple string representation. We examine the design of a trading strategy from two perspec-

tives (a) the complexity of the underlying algorithm and (b) the representation of the underlying time series used in the decision making process. We compare a simple approach based on discrete data to the popular linear SVM with continuous inputs, and introduce a new kernel function that captures the temporal importance of string subsequences. Furthermore we show that this kernel can be evaluated efficiently using a simple weighted averaging process that is equivalent to the Parzen window classifer using that kernel. The algorithms and representations are tested on eight years of foreign exchange data using four different currencies. The results of these experiments suggest that a simple string representation coupled with an averaging process is capable of outperforming more exotic approaches, whilst supporting the idea that when it comes to working in high noise environments often the simplest approach is the most effective.

References

1. Francis EH Tay and Lijuan Cao. Application of support vector machines in financial time series forecasting. *Omega*, 29(4):309–317, 2001.
2. Kyoung-jae Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1):307–319, 2003.
3. Tony Van Gestel, Johan AK Suykens, D-E Baestaens, Annemie Lambrechts, Gert Lanckriet, Bruno Vandaele, Bart De Moor, and Joos Vandewalle. Financial time series prediction using least squares support vector machines within the evidence framework. *Neural Networks, IEEE Transactions on*, 12(4):809–821, 2001.
4. Li-Juan Cao and Francis Eng Hock Tay. Support vector machine with adaptive parameters in financial time series forecasting. *Neural Networks, IEEE Transactions on*, 14(6):1506–1518, 2003.
5. Fernando Perez-Cruz, Julio A Afonso-Rodriguez, and Javier Giner. Estimating garch models using support vector machines. *Quantitative Finance*, 3:163–172, 2003.
6. Altaf Hossain and Mohammed Nasser. Reccurent support and relevance vector machines based model with applications to forecasting volatility of financial returns. *Journal of Intelligent Learning Systems and Applications*, 3:230–241, 2011.
7. Phichhang Ou and Hengshan Wang. Financial volatility forecasting by least square support vector machine based on garch, egarch and gjr models: Evidence from asean stock markets. *International Journal of Economics and Finance*, 2(1):51–64, 2010.
8. Ashraful Islam Khan. Financial volatility forecasting by nonlinear support vector machine heterogeneous autoregressive model: Evidence from nikkei225 stock index. *International Journal of Economics and Finance*, 3(4):138–150, 2011.
9. J. Scott Armstrong, Monica Adya, and Fred Collopy. *Rule-Based Forecasting: Using Judgment in Time-Series Extrapolation.* Kluwer Academic Publishers, 2001.
10. John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis.* Cambridge university press, 2004.
11. Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.
12. Bin Liu, Xiaolong Wang, Lei Lin, Qiwen Dong, and Xuan Wang. A discriminative method for protein remote homology detection and fold recognition combining top-n-grams and latent semantic analysis. *BMC bioinformatics*, 9(1):510, 2008.

Time Series Classification using Compressed Recurrence Plots

Thilo Michael, Stephan Spiegel, Sahin Albayrak

DAI-Lab, Berlin Institute of Technology,
Ernst-Reuter-Platz 7, 10587 Berlin, Berlin
{thmichael, spiegel, albayrak}@dai-lab.de

Abstract. In recent years recurrence plots have become a widely accepted tool for identifying and visualizing structural patterns in time series. It has been shown that the structural patterns found in recurrence plots can be used to determine the similarity between two time series, which is necessary for classification. For instance, it has been proposed to employ video compression algorithms for measuring the similarity between two recurrence plots, which visualize the structural patterns that were extracted from the time series under study.

In this work we assess to what extend the choice of video compression algorithm influences the similarity measurements and classification performance for recurrence plots or time series respectively. Furthermore, we introduce a novel time series distance measure based on the compression of cross recurrence plots. Our evaluation shows that more advanced compression algorithm do not necessarily result in higher classification accuracy, but lead to superior results for relatively long time series.

1 Introduction

Classifying time series has become an important task in many computer science disciplines like data mining and knowledge discovery [4, 12]. An important part of this classification is the definition of a similarity function that measures the distance of two time series under study. In [2] Campana and Keogh proposed a distance measure that employs the video compression algorithm MPEG-1 to determine the similarity of textures. In [10] Silva et al. applied the video compression distance to recurrence plots and showed that especially the classification of shape-based time series benefit from the new approach.

While the MPEG-1 standard proved to be a good basis for the compression of recurrence plots, it is unclear how different and more advanced video encoding alogrithms contained in standards such as MPEG-2 and MPEG-4 effect the distance measure and thus the classification.

In this work we further analyze the possibilities of applying the concept of video compression to recurrence plots. We analyze how well the recurrence plot compression distance performs with newer video compression algorithms such as

MPEG-2 and MPEG-4. We also propose a new compression distances that combines the concepts of video compression distances and cross recurrence plots. In our experimental evaluation we analyze how the different approaches perform with the different video compression algorithms.

2 Background

2.1 Recurrence plots

Recurrence plots (RP) are used to visualize and analyze systems with nonlinear behavior. The input data is usually represented as a vector of real values, which correspond to different states of the system under study. The recurrence plot illustrates recurring states as individual points and recurring segments as line structures [8].

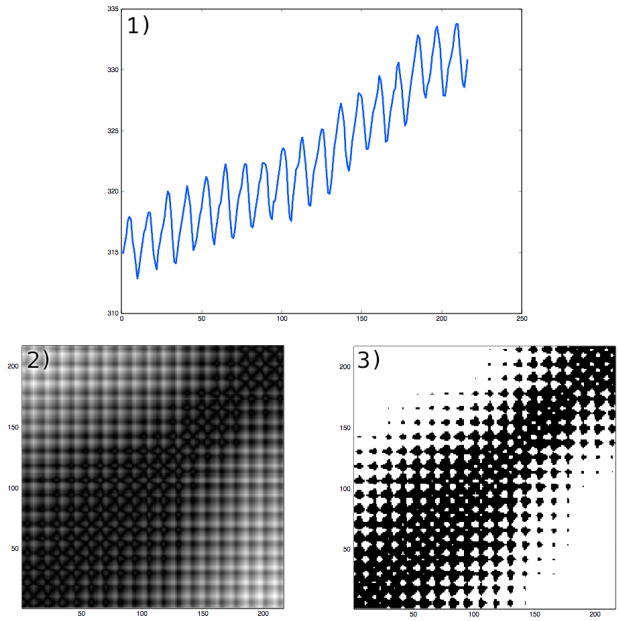


Fig. 1: The plot of a time series (1) showing CO_2 in the atmosphere measured every month between 1958 and 1975 [3]. Below that are the resulting unthresholded recurrnce plot (2) and thresholded recurrence plot with a threshold of 0.2 (3).

A recurrence plots of a time series x can be mathematically described as follows:

$$R_{i,j}^{d,\epsilon} = \Theta(\epsilon - ||x_i - x_j||) \qquad x_i \in \mathbb{R}^d, i, j = 1 \dots n$$

where ϵ is a threshold distance, $||\cdot||$ is a norm, and $\Theta(\cdot)$ is the Heaviside step function. According to this definition, a recurrence of a state at time i at a different time j is pictured within a two-dimensional squared matrix with black and white dots, where black dots mark a recurrence [7].

Unthresholded recurrence plots are a variation where the threshold parameter ϵ and the Heaviside function are removed, which results in gray-shaded plots. Figure 1 compares traditional recurrence plots with unthresholded recurrence plots with the help of a sample time series.

Extending the concept of recurrence plots, cross recurrence plots (CRPs) are used to visualize recurring patterns between two time series x and y:

$$CR_{i,j}^{d,\epsilon} = \Theta(\epsilon - ||x_i - y_j||) \qquad x_i, y_j \in \mathbb{R}^d, i = 1 \dots n, j = 1 \dots m$$

A cross recurrence plot (CRP) shows all those times at which a state in one dynamical systems occurs in a second dynamical system. In other words, the CRP reveals all the times when the trajectories of the first and second time series, x and y, visits roughly the same area in the phase space. The data length, n and m, of both systems can differ, leading to a non-square CRP matrix [9, 11].

2.2 Campana-Keogh distance

The Campana-Keogh distance CK-1 is a Kolmogorov complexity-based compression distance for textures. It uses the MPEG-1 algorithms to calculate a semimetric distance [2].

The MPEG-1 encoding is a specific set of algorithms used to compress a series of coherent images. It not only uses similarities in the horizontal and vertical data of the images (*intra frame compression*) but heavily relies on similarities in temporal space (*inter frame compression*) [6].

When encoding a series of images with MPEG-1, each images is either stored as an intra-frame (often called *I-frame*) or as a predictive frame (often called *P-frame*). The compressed I-frames can be seen as keyframes that contain all the information to reconstruct the images while the P-frames are storing the information that describes the transition from the previous image to the next one. So to decompress a P-frame, the decoding algorithm has to decompress the preceding I-frame and subsequent P-frames up to the current frame [6].

CK-1 makes use of the MPEG-1 I-frames and P-frames to measure the distance between two images. As a prerequisite the two images are converted to grayscale to provide color invariance and are scaled to the same size. Then an MPEG-1 video is created with two frames. The first frame is an intra frame of the first image and the second frame is a predictive frame of the second image. Relying on the inter frame compression of MPEG-1, Campana and Keogh reason that

the size of the compressed *video* is getting smaller as the images get more similar [2].

The CK-1 distances measure is defined as follows:

$$CK1(x,y) = \frac{C(x|y) + C(y|x)}{C(x|x) + C(y|y)} - 1$$

where x and y are the time series to compare and $C(x|y)$ is a function that evaluates the size of a two frame long MPEG-1 file with y as the I-Frame and x as the P-Frame [2].

2.3 Compression distance on recurrence plots

In recent work [10] Silva et al. employed the CK-1 distance to measure the similarity between unthresholded recurrence plots that were generated from time series. They argue that the MPEG video compression algorithm is able to detect similar structures in recurrence plots, which correspond to similar time series patterns. Their experimental evaluation of this recurrence plots compression distance (RPCD) shows that, in comparison to the euclidean distance and dynamic time warping, the combination of the CK-1 distance measure together with unthresholded recurrence plots results in higher classification accuracy for time series which represent shapes (e.g. leaves or faces that are encoded into time series).

3 Proposed method

In the following we introduce our novel cross recurrence plots compression distance (CRPCD), which combines the concept of the discussed compression distance with cross recurrence plots (CRPs). Our proposed approach computes the compression distance between the RPs of two individual time series and their corresponding CRPs.

The intuition behind our proposal is that CRPs are able to reveal co-occurring patterns, which are assumed to be advantageous for video encoding algorithms. Our hypothesis is that the introduced video compression approach produces better classification results when using CRPs, since these plots reveal patterns that co-occur within two time series.

To simplify the notation of the CRPCD, we will denote a recurrence plot R^x with the equivalent cross recurrence plot $CR^{x,x}$.

For the calculation of the cross recurrence plots compression distance of two time series x and y, the plots $CR^{x,x}$, $CR^{y,y}$. $CR^{x,y}$ and $CR^{y,x}$ have to be generated. With a video compression algorithm the distance between $CR^{x,x}$ and $CR^{x,y}$ and

between $CR^{y,y}$ and $CR^{y,x}$ can be calculated as defined by the CK-1 algorithm and then combined to form one distance measure.

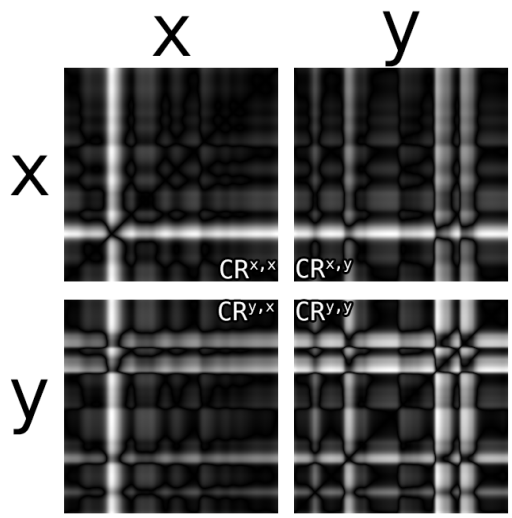


Fig. 2: Cross recurrence plots that are used for the CRPCD measure. On the top are CRPs from x to itself and to y. On the bottom are the CRPs from y to x and to itself.

Mathematically, the CRPCD measure can be defined as follows:

$$CK2(x,y) = \frac{C(CR^{x,x}|CR^{x,y}) + C(CR^{y,y}|CR^{y,x}) + C(CR^{x,y}|CR^{x,x}) + C(CR^{y,x}|CR^{y,y})}{C(CR^{x,y}|CR^{x,y}) + C(CR^{y,x}|CR^{y,x}) + C(CR^{x,x}|CR^{x,x}) + C(CR^{y,y}|CR^{y,y})} - 1$$

where x and y are two time series, $CR^{a,b}$ is the cross recurrence plots of the time series a and b and $C(A|B)$ is a compression function for object A given object B.

4 Experiments

In our experimental evaluation we investigate: i) how the accuracy of RPCD [10] changes when we use other compression algorithms (such as MPEG-2 and MPEG-4); and ii) how our proposed CRPCD measure performs in comparison.

4.1 Experimental setup

We used publicly available libraries [1] to test the RPCD method with the newer compression algorithms MPEG-2 and MPEG-4. We configured the libraries to use modes that favor inter-frame to intra-frame compression, but we did not have the resources to perform an exhaustive search over all possible parameter settings. Hence, there is still potential for further improvement.

Data set	ED	DTW	RPCD MPEG1	RPCD MPEG2	RPCD MPEG4	CRPCD MPEG1	CRPCD MPEG2	Kind
50words	63.10	69.00	77.36	73.85	61.76	**78.46**	71.43	●
Adiac	61.10	60.40	61.64	**72.12**	70.08	61.38	70.08	▼
Beef	53.30	50.00	**63.33**	**63.33**	**63.33**	46.67	46.67	●
ChlorineConcentration	**65.00**	64.80	51.09	63.78	??	48.93	63.33	●
CinC ECG torso	89.70	65.10	**97.90**	97.24	94.20	93.19	86.16	●
Coffee	75.00	82.10	**100.00**	**100.00**	**100.00**	85.71	85.71	●
Cricket X	57.40	**77.70**	70.77	38.46	13.59	75.64	53.59	▲
Cricket Y	64.40	79.20	73.85	42.31	11.03	**82.56**	52.05	▲
Cricket Z	62.00	**79.20**	70.77	40.76	12.82	77.69	56.41	▲
DiatomSizeReduction	93.50	**96.70**	96.41	94.77	95.42	96.08	96.08	●
ECG200	**88.00**	77.00	86.00	87.00	81.00	88.00	86.00	●
ECGFiveDays	79.70	76.80	86.41	87.92	88.27	80.48	**90.36**	●
FaceAll	71.40	80.80	**80.95**	73.96	31.24	80.59	79.74	▼
FaceFour	78.40	83.00	94.32	94.32	90.91	**95.45**	88.64	▼
FacesUCR	76.90	90.49	94.15	78.88	48.00	**95.80**	89.32	▼
Fish	78.30	83.30	87.43	95.43	**96.00**	76.00	93.71	▼
Gun Point	91.30	90.70	**100.00**	**100.00**	**100.00**	98.67	98.00	▲
Haptics	37.00	37.70	38.64	43.83	**44.16**	41.23	43.18	●
InlineSkate	34.20	38.40	32.00	44.00	**45.45**	35.45	42.73	▲
ItalyPowerDemand	**95.50**	95.00	84.26	82.41	69.48	83.77	87.56	●
Lighting2	75.40	**86.90**	75.41	67.21	57.38	81.97	70.49	●
Lighting7	57.50	**72.60**	64.38	21.92	17.81	69.86	42.47	●
MedicalImages	68.40	**73.70**	71.05	61.71	59.60	71.97	68.29	●
Motes	**87.90**	93.50	79.71	65.58	??	82.59	68.37	●
OliveOil	**86.70**	**86.70**	83.33	76.67	83.33	73.33	83.33	●
OSULeaf	51.70	59.10	64.46	83.06	**83.88**	65.29	80.17	▼
RobotSurface	69.50	72.50	**79.70**	70.55	66.39	79.70	73.04	●
RobotSurfaceII	**85.90**	83.10	84.26	78.49	51.31	84.47	84.26	●
SwedishLeaf	78.70	79.00	**90.24**	88.64	86.24	88.80	88.48	▼
Symbols	90.00	95.00	90.45	**96.38**	96.08	90.05	96.18	●
WordsSynonyms	61.80	64.90	72.41	71.00	62.54	**73.35**	67.24	●
Wins	6/31	7/31	7/31	5/31	7/31	5/31	1/31	

Table 1: Accuracy rates for each distance measure. Following the notation in [10], the symbols ▼, ▲ and ● represent data sets generated from figure shapes, human movements and all remaining data sets, respectively.

For the CRPCD approach with the MPEG-1 video encoding algorithm we used the test setup described in [10] but implemented our new compression distance approach to preclude distorted results due to different configurations of the video encoding algorithms.

In the evaluation we used the nearest neighbor method to classify the data sets provided by the UCR Time Series Classification/Clustering Page [5]. Due to the fact that we had four different distance measures to test, two of which were based on the CRPCD

method that requires 8 video compressions to be made, we removed the largest data sets from the testing corpus. The results of the evaluation can be seen in table 1.

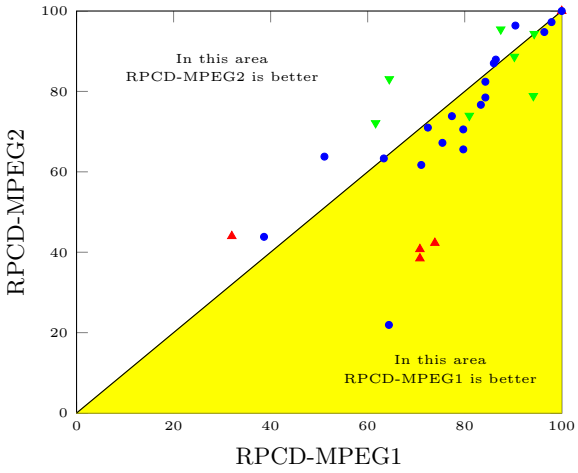


Fig. 3: A plot comparing the accuracy of the RPCD method using MPEG-1 (bottom-right) and MPEG-2 (top-left) video encoding algorithms. Each point represents a data set. In all data sets in the white area of the plot the MPEG-2 approach outperformed MPEG-1. Following the notation in [10], the symbols ▼, ▲ and ● represent data sets generated from figure shapes, human movements and all remaining data sets, respectively.

4.2 Results and Discussion

As seen in Table 1, each tested setup, except for the CRPCD-MPEG2 approach, had roughly the same number of data sets where it scored the best result (values in bold). This shows that the different measurement approaches complement each other in a way that none of the tested measures outperform all the others.

Notably large time series like InlineSkate (1882 data points) or Haptics (1092 data points) performed better with the newer video compression algorithms. We believe that this is due to the fact that with large plots the overhead and the intra-frame encoding are less significant. Thus the resulting file size depends more on the inter-frame compression of two images.

The approach to calculate the RPCD measure with the newer video encoding algorithm MPEG-2 showed no significant improvements to the RPCD implementation as can be seen in Figure 3. Notably, most of the data sets that were generated by human movement performed significantly worse. Because the MPEG-2 algorithm is used as a *black box* it is difficult to identify the reasons of this unexpected result. Possible reasons may be due to the fact that the MPEG-2 video encoding produces a bigger file overhead and that the intra-frame compression improved over the MPEG-1 algorithm. This may lower the significance of the inter-frame compression in the video file.

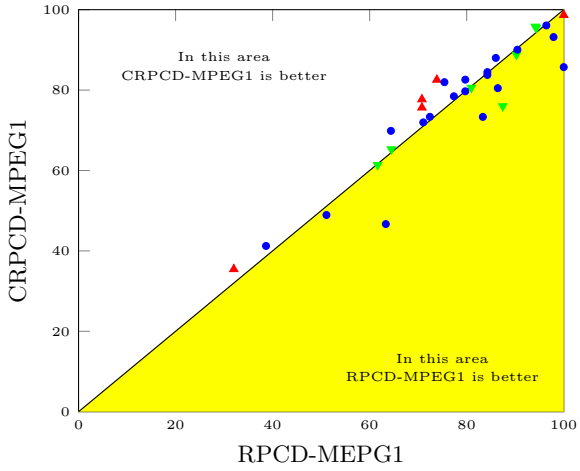


Fig. 4: A plot comparing the accuracy of the RPCD measure to the CRPCD measure - both with the MPEG-1 video encoding algorithm - where each point represents a data set. In all data sets in the white area of the plot CRPCD outperformed RPCD. Following the notation in [10], the symbols ▼, ▲ and ● represent data sets generated from figure shapes, human movements and all remaining data sets, respectively.

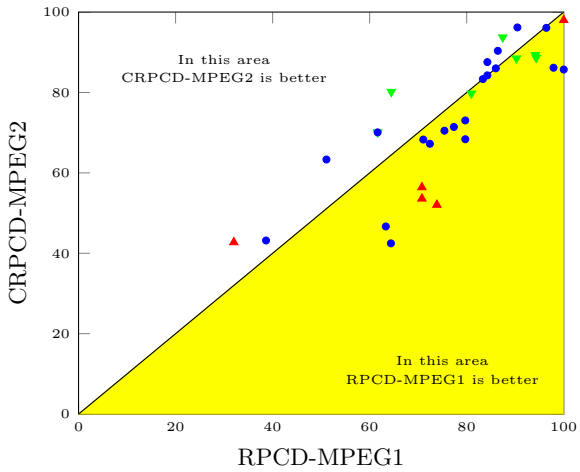


Fig. 5: A plot comparing the accuracy of the RPCD measure with MPEG-1 video encoding to the CRPCD measure with MPEG-2 video encoding, where each point represents a data set. In all data sets in the white area of the plot CRPCD-MPEG2 outperformed RPCD-MPEG1. Following the notation in [10], the symbols ▼, ▲ and ● represent data sets generated from figure shapes, human movements and all remaining data sets, respectively.

The CRPCD method tested with the MPEG-1 algorithm shows slight improvements to the original RPCD approach. Especially time series that were generate by human movement improved in contrast to RPCD as can be seen in Figure 4. However it is notably that the calculation time of the CK-2 algorithm is significantly higher than the time of the RPCD method because our approach requires two times as many plots to be generated and videos to be encoded.

We tested the newer video encoding algorithm MPEG-2 with the CRPCD approach combined, to see how these two factors influence each other. Apart from a few outliers, the MPEG-2 CRPCD method performs worse than the MPEG-1 CRPCD approach, as can be seen in Figure 5. We reason that in line with our hypothesis that the MPEG-2 algorithm undermines the inter-frame compression with the intra-frame compression, the CRPCD method further lessens the significance of the compression between two plots.

Additionally to the MPEG-2 video encoding algorithm we also tested the more advanced MPEG-4 algorithms. Overall the data sets that performed well with the MPEG-2 algorithm also performed well with the MPEG-4 video encoder and data sets that had less accuracy with MPEG-2 also had less accuracy with MPEG-4.

5 Conclusion

In this work we evaluated the use of more advanced video encoding algorithms like MPEG-2 and MPEG-4 with the RPCD measure. We furthermore introduced a new recurrence plot-based distance measure CRPCD that utilizes cross recurrence plots.

We showed that more advanced compression algorithms do not necessarily result in higher classification accuracy. However, we observed that the MPEG-2 and MPEG-4 algorithms yield better performance for relatively long time series. Though none of the classifiers is consistently better, the results show that they complement one another.

In future work we intend to evaluate the correlation between the size of a time series and the accuracy of video compression-based distance measures. Future work may also include the modification of video encoding algorithms in a way that the compression only takes place in temporal space so that overhead and unwanted intra-frame-compression get minimized.

Bibliography

[1] F. Bellard, M. Niedermayer, et al. Ffmpeg. *Available from: http://ffmpeg.org*, 2012.

[2] B. J. Campana and E. J. Keogh. A compression-based distance measure for texture. *Statistical Analysis and Data Mining*, 3(6):381–398, 2010.

[3] D. Kahaner, C. Moler, and S. Nash. Numerical methods and software. *Englewood Cliffs: Prentice Hall, 1989*, 1, 1989.

[4] E. Keogh, S. Lonardi, C. A. Ratanamahatana, L. Wei, S.-H. Lee, and J. Handley. Compression-based data mining of sequential data. *Data Mining and Knowledge Discovery*, 14(1):99–129, 2007.

[5] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana. The ucr time series classification/clustering homepage. *URL= http://www. cs. ucr. edu/˜ eamonn/time_series_data*, 2006.

[6] D. Le Gall. Mpeg: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, 1991.

[7] N. Marwan. *Encounters with neighbours: current developments of concepts based on recurrence plots and their applications*. Norbert Marwan, 2003.

[8] N. Marwan, M. Carmen Romano, M. Thiel, and J. Kurths. Recurrence plots for the analysis of complex systems. *Physics Reports*, 438(5):237–329, 2007.

[9] N. Marwan and J. Kurths. Cross recurrence plots and their applications. *Mathematical physics research at the cutting edge*, pages 101–139, 2004.

[10] D. F. Silva, V. Souza, M. De, and G. E. Batista. Time series classification using compression distance of recurrence plots. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 687–696. IEEE, 2013.

[11] S. Spiegel, J.-B. Jain, and S. Albayrak. A recurrence plot-based distance measure. In *Translational Recurrences*, pages 1–15. Springer International Publishing, 2014.

[12] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.

Keyword Index

Adaptive ensemble size 38
Audio Signal Analysis 110
Automatic indexing 146

Business Process 74

Centroid classier 134
Change Mining 50
Class imbalanced data 62
Classification 74, 178
Community detection 26
Complex distributions of the minority class 62
Complex Dynamic Data 50
Cross-domain generalization 98

Data mining 146
Data Mining 14
Data streams 2, 14, 38

Ensemble classifiers 62
Event stream 26

feature construction 166
Financial prediction 166
Frequent Itemset Mining 14
Frequent subgraph mining 122

Graph 156
Graph kernel 122

Heterogeneous information networks 134

Intelligent Transport System 110

Knowledge discovery 86

Logic programming 98

Metalearning 38
Multi-target regression 2

Network analysis 134
Network decomposition 134

OzaBag 38

PageRank 134
Parallel Algorithms 14
Parzen windows 166
Periodicity 50
Predictive clustering trees 86
Process mining 26

Reasoning by analogy 98
Reconfigurable Hardware 14
Recurrence Plots 178
Redescription mining 86
Relational databases 146
Relational Patterns 50
Return on Investment 38
Roughly Balanced Bagging 62

Security 74
Sequential data 156
Sequential pattern mining 146
support vector machines 166
SVM 134

Text mining heuristics 134
Time Series 178
Transductive learning 156
Tree mining 122
Tree-based approaches 2

Uncertainty 74

Vehicle classification 110
Video Compression 178

World trade 86

Author Index

Albayrak, Sahin	178
Appice, Annalisa	26
Bande Serrano, José Manuel	14
Bustio, Lazaro	14
Cousins, Simon	166
Cumplido, René	14
Di Pietro, Marco	26
Dzeroski, Saso	2
Džeroski, Sašo	86
Fazzinga, Bettina	74
Feregrino, Claudia	14
Ferilli, Stefano	98
Flesca, Sergio	74
Funaya, Koichi	156
Furfaro, Filippo	74
Greco, Claudio	26
Hernandez Leon, Raudel	14
Horvath, Tamas	122
Kralj, Jan	134
Kubera, Elzbieta	110
Lango, Mateusz	62
Lavrač, Nada	86, 134
Leuzzi, Fabio	98
Loglisci, Corrado	50
Malerba, Donato	26, 50
Mazuran, Mirjana	146
Michael, Thilo	178
Mihelčić, Matej	86
Olorunnimbe, M. Kehinde	38
Osojnik, Aljaž	2
Panov, Pance	2
Paquet, Eric	38
Pontieri, Luigi	74

Robnik-Šikonja, Marko 134

Simoni, Matteo 146
Skrzypiec, Krzysztof 110
Slowik, Tomasz 110
Smuc, Tomislav 86
Spiegel, Stephan 178
Stefanowski, Jerzy 1, 62

Tanca, Letizia 146

Viktor, Herna 38

Welke, Pascal 122
Wieczorkowska, Alicja 110
Wrobel, Stefan 122

Xu, Zhao 156

Zlicar, Blaz 166