Proceedings of the
ACM SIGKDD Workshop on
Useful Patterns

UP '10

www.usefulpatterns.org

A full-day workshop in conjunction with
the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining
in Washington, DC, USA, on 25 July 2010.

Jilles Vreeken & Nikolaj Tatti & Bart Goethals, editors

Proceedings of the ACM SIGKDD Workshop on Useful Patterns

These proceedings are also included in the ACM Digital Library.

UP'10, July 25, 2010, Washington, DC, USA

ACM SIGKDD Workshop on Useful Patterns

General Chairs

Bart Goethals & Nikolaj Tatti & Jilles Vreeken (Universiteit Antwerpen)

Program Committee

Michael Berthold (Universität Konstanz)
Björn Bringmann (Katholieke Universiteit Leuven)
Johannes Fürnkranz (Technische Universität Darmstadt)
Vivekanand Gopalkrishnan (Nanyang Technological University)
Ruoming Jin (Kent State University)
Eamonn Keogh (University of California – Riverside)
Arno Knobbe (Universiteit Leiden)
Arne Koopman (Universiteit Leiden)
Carson K. Leung (University of Manitoba)
Srinivasan Parthasarathy (Ohio State University)
Jian Pei (Simon Fraser University)
Kai Puolamäki (Aalto University)
Geoff Webb (Monash University)

External Reviewer

Ardian Kristanto Poernomo

Preface

Pattern mining is an important aspect of data mining, concerned with finding local structure in data. Traditionally, the focus of research in pattern mining has been on completeness and efficiency. That is, trying to find all potentially interesting patterns as fast as possible. This focus, important as it is, has led our attention away from the most important aspect of the exercise: leading to useful results. Let's consider the following example.

> Say a domain expert wants to extract novel knowledge from some data, and specifically wants to know what patterns are present in the data. To do so, the expert involves a data analyst. The analyst is provided with the data, and runs his favorite pattern mining algorithm. Due to the pattern explosion, the number of discovered patterns the analyst will find will be enormous; the result of the mining exercise often being much larger than the original data. Nevertheless, let us assume our expert patiently considers the result. Although he might stumble upon some interesting patterns, he will mostly encounter very many patterns that convey roughly the same information. Perhaps worse, however, is that he will find that many of the patterns represent information that is already known.

All things considered, even when convinced of the potential, in the above case the expert would not be very impressed by the usefulness of pattern mining. Unlike in other fields of data mining, such as clustering, in pattern mining presentation and visualization has not been a priority. However, even when we forget about presentation to a user, patterns are not yet as useful as they could be. While they provide highly detailed descriptions of phenomena in data, it remains difficult to make good use of them in, say, e.g., classification or clustering. While this is mostly due to the huge number of discovered patterns, making the result unwieldy at best, it does pose interesting research questions like 'how to select patterns such that they are useful?'. Techniques that summarize the result exist, but focus primarily on being able to reconstruct the full set, instead of targeting the usability of the summarized set. As such, research into techniques that mine small sets of high-quality patterns is required, where high-quality is directly related their intended use.

It is exactly this research, experiences and practices that we want to discuss with UP. The main program of UP'10 consists of eight papers covering various aspects of useful pattern mining. We sincerely thank the authors of the submissions and the attendees of the workshop. We wish to thank the members of our program committee for their help in selecting a set of high-quality papers. Furthermore, we are very grateful to Jiawei Han and Geoff Webb for giving keynote presentations about their recent work on useful patterns.

Bart Goethals & Nikolaj Tatti & Jilles Vreeken

Antwerp, May 2010

Table of Contents

Invited Talk

Mining Useful Patterns: My Evolutionary View

Jiawei Han
Department of Computer Science
University of Illinois at Urbana-Champaign
hanj@cs.uiuc.edu

Abstract

Pattern mining has been studied in the data mining community for over 15 years, with lots of interesting results and methods reported. One critical issue in pattern mining is the usefulness of patterns, i.e., what patterns are likely useful for what applications, instead of yet another efficient pattern mining algorithm. In my talk, I will discuss my evolutionary view on the usefulness of patterns and present a set of examples on what patterns are considered to be useful in certain practice. This may give some insight on pattern analysis, based on my own study, and point out a few open research problems and possible exploration of broad applications of pattern mining.

Bio

Jiawei Han, Professor of Computer Science, University of Illinois at Urbana-Champaign. He has been researching into data mining, information network analysis, database systems, and data warehousing, with over 450 journal and conference publications. He has chaired or served on many program committees of international conferences, such as PC co-chair for KDD, SDM, and ICDM conferences.

He is currently the founding Editor-In-Chief of ACM Transactions on Knowledge Discovery from Data and as the Director of Information Network Academic Research Center supported by U.S. Army Research Lab. He is a Fellow of ACM and IEEE, and received 2004 ACM SIGKDD Innovations Award, 2005 IEEE Computer Society Technical Achievement Award, and 2009 IEEE Computer Society Wallace McDowell Award. His book "Data Mining: Concepts and Techniques" (2nd ed., Morgan Kaufmann, 2006) has been adopted as a textbook worldwide.

UP'10, July 25, 2010, Washington, DC, USA.
ACM 978-1-4503-0216-6/10/07.

Invited Talk

Association Discovery

Geoffrey I. Webb
Faculty of Information Technology
Monash University, Australia
Geoff.Webb@monash.edu

Abstract

Association discovery is one of the most studied tasks in the field of data mining. However, far more attention has been paid to how to discover associations than to what associations should be discovered. In this talk Geoff will provide a highly subjective tour of the field. He will highlight shortcomings of the dominant frequent pattern paradigm and illustrate benefits of the alternative top-k approach. He will argue for the value of statistical filtering of associations and discuss some null-hypotheses of widespread application. He will compare the merits of randomization, holdout and within-search approaches to statistical filtering. He will also argue that in many applications it is preferable to find interesting itemsets rather than interesting rules.

Bio

Geoff Webb, Professor of Computer Science, Director of the Centre for Research in Intelligent Systems, Monash University. His primary research areas are machine learning, data mining and user modeling, with over 150 journal and conference publications. He is the author of the commercial data mining package Magnum Opus, a system that embodies many of his research contributions in data mining. Many of his learning algorithms are included in the widely-used Weka machine learning workbench.

He is currently Editor-In-Chief of the Springer journal Data Mining and Knowledge Discovery, co-editor of the Springer Encyclopedia of Machine Learning, member of the editorial boards of ACM Transactions on Knowledge Discovery from Data and Machine Learning, and member of the editorial advisory board of Statistical Analysis and Data Mining.

UP'10, July 25, 2010, Washington, DC, USA.
ACM 978-1-4503-0216-6/10/07.

Patterns from multiresolution 0-1 data

Prem Raj Adhikari and Jaakko Hollmén
Aalto University School of Science and Technology
Department of Information and Computer Science
PO Box 15400, FI-00076 Aalto
prem.adhikari@tkk.fi and jaakko.hollmen@tkk.fi

ABSTRACT

Biological systems are complex systems and often the biological data is available in different resolutions. Computational algorithms are often designed to work with only specific resolution of data. Hence, upsampling or downsampling is necessary before the data can be fed to the algorithm. Moreover, high-resolution data incorporates significant amount of noise thus producing explosion of redundant patterns such as maximal frequent itemset, closed frequent itemset and non-derivable itemset in the data which can be solved by downsampling the data if the information loss is insignificant during sampling. Furthermore, comparing the results of an algorithm on data in different resolution can produce interesting results which aids in determining suitable resolution of data. In addition, experiments in different resolutions can be helpful in determining the appropriate resolution for computational methods. In this paper, three methods of downsampling are proposed, implemented and experiments are performed on different resolutions and the suitability of the proposed methods are validated and the results compared. Mixture models are trained on the data and the results are analyzed and it was seen that the proposed methods produce plausible results showing that the significant patterns in the data are retained in lower resolution. The proposed methods can be extensively used in integration of databases.

Keywords

Binary data, multiple resolutions, Upsampling, Downsampling, Mixture Models

1. INTRODUCTION

This paper proposes and studies three different downsampling methods for genome-wide chromosome bands in amplification data. Sample in this context is a process of defining the level of precision for staining the chromosome bands. For example, chromosome-1 can be defined by 23, 28, 42, 61 and 63 bands in resolution 300, 400, 550, 700 and 850 respectively as defined by International System on Cytogentic Nomenclature(ISCN)[1]. The proposed methods can also be used in downsampling of similar data where the data is encoded in different chromosome bands for each sample. Biological systems are very complex systems. Since these complexities are directly related to the health of humans, different technologies have been developed to study them. Microarray technology, such as CGH (Comparative genomic hybridization)[2] and aCGH (Array comparative genomic hybridization) [3] have given the facilities to study the genomes and the genes in human body. Thus, biological data are available in different resolutions. However, computational algorithms can often handle only specific resolution of the data. So, data needs to be upsampled and downsampled to different resolutions before some algorithms are applied on them. Furthermore, comparing the results of the models in different resolutions can reveal some interesting facts useful for cancer research. If data in lower resolution produces results comparable to data in higher resolution, the time, computational and hardware costs required to obtain the data in higher resolution can be saved. Here, a dataset in resolution 850 was downsampled in four different resolutions 300, 393, 550, and 700 and experiments were performed on the data in different resolutions. In addition, a different dataset in resolution 393 was upsampled to resolution 550, 700 and 850 and downsampled to resolution 300. Other popular dimensionality reduction methods[4] does not produce desirable results because representation of the data is lost. In this context, we present methods of upsampling and downsampling of data and experimental results in integrating two biological databases originally in different resolutions.

The major aim of the paper is to sample the data in different resolutions such that the significant patterns i.e. frequent itemsets are retained in the sampled data. Thus, we experiment our proposed methods with a set of pattern mining algorithms. Given a binary data, $\mathcal{D}$ with a set of attributes $\mathcal{I}_1, \mathcal{I}_2 \ldots \mathcal{I}_n$ and a support σ, frequent set is the set $\mathcal{F}$ of items of $\mathcal{D}$ such that at least a fraction of σ of the rows of $\mathcal{D}$ have 1 in all columns of $\mathcal{F}$ [5, 6]. However, the major problem with frequent itemset is that if an itemset $\{a, b, c\}$ is frequent then their subsets are also frequent because of the anti-monotonicity property of frequent itemsets[7] thus making it unsuitable for comparison and reporting. On the other hand, maximal frequent itemset can be defined as an itemset which is frequent but non of its supersets are frequent [8]. Hence, we experiment our sampling methods with maximal frequent itemset.

UP'10, July 25th, 2010, Washington, DC, USA.

The rest of the paper is organized as follows. Section 2 briefly surveys the literature and Section 3 gives some brief information about the dataset used in these experiments. Section 4 and 5 discuss the methods used in upsampling and three methods used in downsampling. Section 6 summarizes the details of model selection procedure in the context of mixture models. Section 7 explains the implementation of the proposed methods, discusses the experiments performed on different methods, and also compares results of the experiments on different methods. In Section 8, conclusion is drawn from the experiments. Finally, Section 9 gives the future directions for research and issues not covered in this paper.

2. RELATED WORK

DNA copy number analysis was started in [9] where the authors mainly focused on determining the copy number of the cytogenetic band. Similar works performed were reviewed in [10] to determine the copy number. However, in [9] and [10], the authors did not establish a relation between the copy number and their clinical significance. In the recent past, DNA copy number amplification data collected with bibliomics survey from 838 journal articles published from 1992 to 2002 was analyzed in [11]. In the work, amplification patterns were determined for 73 different neoplasms and the neoplasms were clustered according to amplification profiles thus identifying the amplification hotspots using independent component analysis(ICA). The profiling revealed that human neoplasms formed clustered based on the amplification frequency. Continuing the studies in DNA copy number amplification, authors in [12] classified the human cancers based on copy number amplification using probabilistic modelling. Furthermore, the authors extracted the ranges of amplification in the chromosome and expressed it according to the cytogenetic nomenclature. In [13] and [14], the authors modeled the DNA copy number amplification using a mixture of multivariate Bernoulli Distribution. The classification of 73 different neoplasms in [11] were extended to 95 different neoplasm types. In [14] authors have proposed a compact and understadable reprsentation of the multivariate Bernoulli mixture model. Furthermore, in [15], the authors have proposed the enhancement to Bayesian Piecewise Constant Regression(BPCR) called mBPCR changing the segment number estimator and boundary estimator to enhance the fitting procedure. The proposed mBPCR was more accurate in the determination of true breakpoints of amplification. The more recent studies [16] and [17] have mainly focused in cancer specific analysis of DNA copy number. Although the mixture models were used in [13] and [14], they have studied only chromosome-1 data in resolution 393. Chromosome-1 being the largest chromosome, there are significant amount of amplifications [11]. However, a single chromosome band and the specific gene responsible for cancer has not been identified. Hence, study was performed on all chromosomes including chromosome-1. Chromosomewise analysis can reveal interesting facts about amplification of specific chromosomes and guarantees efficient computation & ease of analysis. Furthermore, there are several sources of multilevel biological data that comes in multiple resolutions but there seems to be a significant gap in research to deal with multiple resolution of the data. Algorithms and methods to deal with such multi-resolution data could possess very high clinical significance.

3. DATASET

The dataset provided was a binary (0-1) dataset about DNA amplifications specifying amplification of certain band of chromosome. DNA copy number amplifications are mutations in the DNA structure. The data was collected by bibliomics survey of 838 journal articles during 1992-2002 by hand without using state-of-the-art text mining techniques [12, 14]. The dataset contained the information about the amplification patterns of 4590 cancer patients. Each row describes one sample of cancer patient while each column identifies one chromosomal band(region). The amplified chromosomal regions were marked with 1 while and the value 0 defines that the chromosome band is not amplified. Chromosomes X and Y were not included in the experiments because of the lack of data. Patients whose chromosomal band had not shown any amplification for specific chromosome were not included in the experiments Thus different chromosomes had different number of samples.

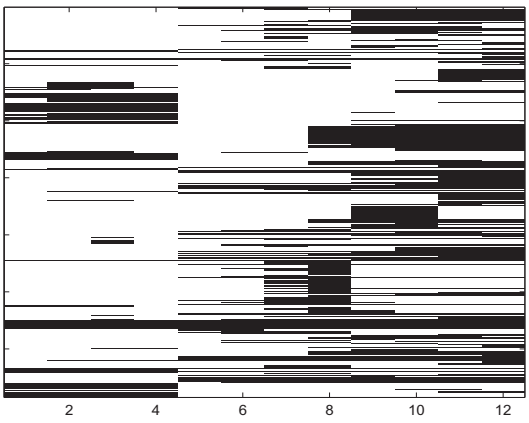


Figure 1: ***DNA copy number amplifications in chromosome-17, resolution 393.*** $\chi = (X_{ij})$, $X_{ij} \in \{0, 1\}$ **. Each row represents one sample of the amplification pattern for a patient and each column represents one of the chromosome bands.**

The data for chromosome-17 in resolution 393 demonstrated in the Figure 1, the copy number amplifications occur very sparsely and are often skewed. The original data was in the resolution 400 i.e. there were 393 chromosomal bands(regions) for the entire genome. The original data was upsampled to resolution 550, 700 and 850 and downsampled to resolution 300 using the methods discussed in Section 5. Bands for specific chromosome were extracted and mixture modelling was preformed on each chromosome. For example: chromosome-1 had 63, 61, 42, 28, and 23 chromosomal bands in resolution 850, 700, 550, 400, and 300 respectively [1]. Similarly, a different set of data was available in resolution 850. The data in resolution 850 was different than that in the resolution 400. Similar to the data in the resolution 400, the data in resolution 850 was downsampled to resolution 300, 400, 550 and 700. Element-wise AND operation over all the samples in the data results in a zero vector which necessitates sophisticated machine learning and data mining methods and techniques for classifying and profiling amplification.

4. UPSAMPLING

Upsampling is the process of changing the representation of data to the higher or finer resolution. A simple method was devised to upsample the data from resolution 393 and three different methods were used to downsample the data from higher resolution. Upsampling was simple and were implemented using simple transformation tables. Initially, the dataset was in resolution 393 and it was upsampled to three different resolutions 550, 700 and 850. A simple method was used to upsample the data. Multiple copies of cytogenetic band in lower resolution were made to upsample the data to higher resolution. For example, cytogenetic band 1q36.1 in resolution 550 has been divided into three bands 1q36.11, 1q36.12 and 1q36.13 in resolution 850. So, multiple copies of 1q36.1 was made for all bands 1q36.11, 1q36.12 and 1q36.13 in resolution. Figure 2 depicts the process of upsampling.

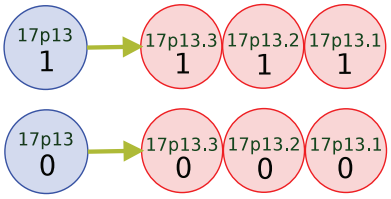


Figure 2: ***Schematic representation of upsampling where duplicate copies of similar cytogenetic bands are made in the higher resolution.***

Figure 2 shows that three copies of similar cytogenetic band in lower resolution band are made to upsample the data to higher resolution. When multiple copies of same cytogenetic band is made higher resolution will have only few unique rows. Hence, when the sample size decreases the complex model in higher dimension can not be trained to convergence thus producing poor results. Implementation of downsampling was performed using simple transformation tables implemented in Perl[18]. Table 1 shows an example of table for transformation of data in 393 resolution to 850 resolution for chromosome 17.

Table 1 shows that some chromosome bands missing in 393 resolution are seen in resolution 850. Hence, duplicate copies of the similar chromosome band in resolution 393 were made in higher resolution. Duplications were based on the assumption that if an adjacent area is amplified then the probability of the chromosome band being amplified is high because amplifications typically cover large areas. The transformation table were chromosome specific and resolution specific (i.e. 88 transformation table in all for different chromosomes)

5. DOWNSAMPLING

Downsampling is the process of changing the representation of the data to the lower or coarser resolution. In both cases of upsampling and downsampling no attempt is made to infer the structure of the data and no information is added or removed during the process. If the data of the same patients were available in two different resolutions, one of the supervised classification algorithm machine learning could be used in downsampling. However, such data was not available and hence simple but useful methods are used for downsampling. Downsampling methods were implemented in scripts with a script for each chromosome in each resolution. Section 5.1, 5.2 and 5.3 detail the methods of downsampling. Interestingly, in some cases there were some cytogenetic bands which were not available in higher resolution. For instance, the q arm of chromosome-4 in resolution 850 is divided into 4q35.1 and 4q35.2. In contrast, in resolution 700, the q arm of chromosome -4 is divided into three bands: 4q35.1, 4q35.2 and 4q35.3. respectively. In such cases, missing band in lower resolution was assigned the amplification pattern of its nearest neighbor in all three methods. For the example case above, the cytogenetic band 4q35.3 was assigned the amplification pattern of 4q35.2.

Resolution 393	Resolution 850
17p13	17p13.3
...	17p13.2
...	17p13.1
17p12	17p12
17p11.2	17p11.2
17p11.1	17p11.1
17q11.1	17q11.1
17q11.2	17q11.2
17q12	17q12
17q21	17q21.1
...	17q21.2
...	17q21.31
...	17q21.32
...	17q21.33
17q22	17q22
17q23	17q23.1
...	17q23.2
...	17q23.3
17q24	17q24.1
...	17q24.2
...	17q24.3
17q25	17q25.1
...	17q25.2
...	17q25.3

Table 1: Chromosome bands for resolution 393 & 850 and their transformation.

5.1 OR-function Downsampling

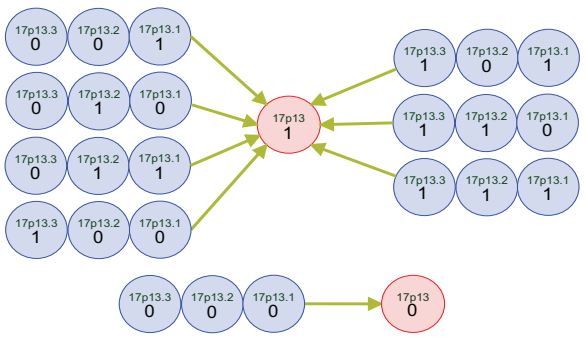


Figure 3: Schematic representation of OR-function downsampling procedure. Here the cytogenetic band in lower resolution is amplified if any of the bands in higher resolution is amplified. Cytogenetic band in lower resolution is not amplified only when none of the bands in higher resolution is amplified.

In OR-function downsampling method, the cytogenetic band in lower resolution is not amplified if none of the bands in higher resolution are amplified. The cytogenetic band in lower resolution is amplified if either of the bands in higher resolution is amplified. Figure 3 depicts the OR-function downsampling method. The OR-function downsampling method is based on simple belief that if the one of the bands in higher resolution is amplified, it signifies the presence of amplification in the band. For the case in the Figure 3 downsampling can be considered as a simple binary classification problem in machine learning where input is three dimensional binary variable and output is one dimensional binary variable. The solution is a simple truth table describing the classical OR operation.

5.2 Majority decision Downsampling

In majority decision downsampling method, a cytogenetic band in lower resolution is amplified if majority of the cytogenetic bands in higher resolution are amplified otherwise the cytogenetic band is not amplified. In case of a tie amplification of two nearest bands one in the left and other one in the right are taken into consideration iteratively and the amplification pattern of the band is determined using idea similar to 'golden goal'[1] strategy used in football. In other words, if in any iteration both bands in neighborhood bands are amplified than the band is amplified and if both the neighbors are unamplified than the band is deemed unamplified. If the amplification of lower resolution can not be concluded with 'golden goal' strategy then the band in lower resolution is deemed as amplified. Figure 4 shows one of the examples of majority decision in downsampling.

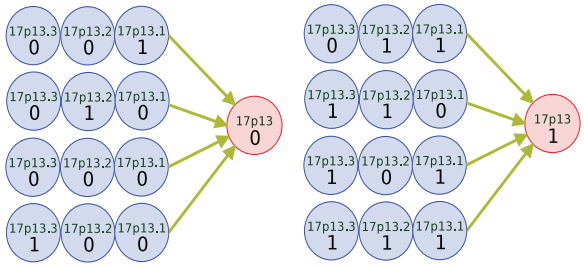


Figure 4: Schematic representation of majority decision downsampling procedure. Here the cytogenetic band in lower resolution is amplified if majority of the bands in higher resolution are amplified, otherwise it not amplified.

There is a shortcoming in this downsampling procedure because the majority decision downsampling procedure does not take into account the the lengths of the cytogenetic bands. The lengths of cytogentenic bands are considered by length weighted downsampling method discussed in Section 5.3.

5.3 Length weighted Downsampling

As shown in the Figure 5, length weighted downsampling method considers the length of the cytogenetic band. The length of the cytogenetic band varies in each assembly and hence relative lengths were considered. The amplification of cytogenetic band in lower resolution is determined by the weighted length of cytogenetic band in higher resolution. Each cytogenetic band is weighted according to the relative length of the cytogenetic band. If the total length of amplified region is greater than the total length of unamplified region, the cytogenetic band in lower resolution is amplified, otherwise the cytogenetic band is unamplified. Here, relative length is considered which gives more accurate measure of the amplification profiles in the cytogenetic band. Absolute lengths of the cytogenetic bands are not currently available and vary with each assembly. Two relative measures were considered in the calculation of the length. From the ideogram dataset available in NCBI [19], the difference between ISCN.top and ISCN.bot were used as relative measures. Similarly, difference between bases-top and bases-bot were also used as the relative measure of the length of each cytogenetic band. The difference in the results produced using the different relative measure of length have also been studied.

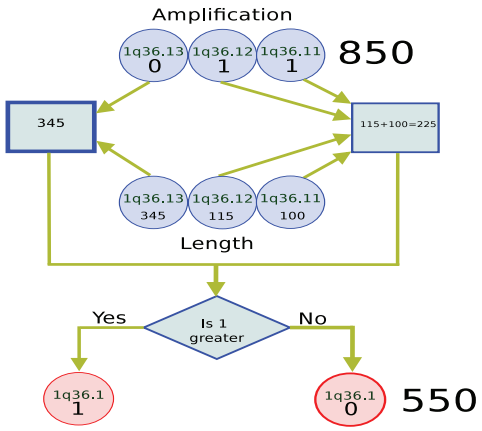


Figure 5: Schematic representation of weighted average downsampling procedure. Here the cytogenetic band in lower resolution is amplified if total length of the amplified bands in higher resolution is greater than the total length of unamplified bands, otherwise it not amplified. The figure is an example case in chromosome 1q36.1 where two cytogenetic bands 1q36.11 and 1q36.12 in resolution 850 are amplified and one band 1q36.13 is not amplified. However, total length of unamplified region i.e. band 1q36.13 (345) is greater than total length of the unamplified region i.e. bands 1q36.11 and 1q36.12 (100+115=225). Hence, the band in resolution 550 is unamplified.

6. MODEL SELECTION

Cancer is not a single disease but a collection of diseases. Furthermore, cancer is a multi-factorial[2] disease. Therefore, finite mixture models [20, 21, 22] was selected to model the amplification data because they provide efficient method to

[1]The golden goal is a method used in football to determine the winner which end in a draw after the end of regulation time. Golden goal rules allow the team that scores the first goal during extra time to be declared the winner. The game finishes when a golden goal is scored.

[2]Here multi-factorial is used to mean there are many factors causing cancer. Majority of the noninfectious diseases are multi-factorial.

model the heterogeneous population. Furthermore, since the copy number amplification data was a high dimensional binary data, the distribution used in the mixture model is Bernoulli distribution. Assuming that the data comes from a mixture of known number of components, J, finite mixture of multivariate Bernoulli distributions is defined as:

$$p(\mathcal{D}|\Theta) = \sum_{j=1}^{\mathrm{J}} \pi_j \prod_{i=1}^{\mathrm{d}} \theta_{ji}^{x_i} (1-\theta_{ji})^{1-x_i} \quad (1)$$

where π_j are the mixture proportions satisfying the properties such as convex combination such that $\pi_j \geq 0$ and $\sum_{j=1}^{\mathrm{J}} \pi_j = 1$ for all $j = 1, \ldots J$. Θ is composed of $\theta_1, \theta_2, \theta_3 \ldots \theta_d$ for each component distribution. Selection of number of mixture components J directly influences the performance of the mixture models. With fewer number of components, the mixture model behaves similar to a parametric model and increases the bias. On the contrary, if the mixture model has a large number of components then the model can overfit the data thus producing unreasonable variation. Hence, there is always a trade-off between the two. To optimize the trade-off and determine optimal number of components in the mixture model, 10-fold cross-validation technique [23, 24] was used. Expectation Maximization (EM) algorithm [25, 26] was used to train the mixture model using BernoulliMix programme package [27] freely available in BernoulliMix homepage. The model selection approach used in the paper is similar to [13, 14] except for the cross-validation procedure.

7. EXPERIMENTS

The downsampling methods were implemented in scripts, one each for each method, each chromosome and each resolution. Chromosomes X and Y were excluded from the experiments because of the lack of data. Hence, there were 198 scripts in all for all transformations. Matlab ®[28] was used for scripting. The individual scripts for downsampling each chromosome takes a file name of the data set in higher resolution as input checks for the abnormality in the data. The data was then transformed band-wise to lower resolution combining the multiple bands in higher resolution according to the three different methods proposed in Sections 5.1, 5.2 and 5.3. Furthermore, samples which contained no amplifications were also removed from the data.

7.1 Comparison of Downsampling Methods

The downsampled data from 850 resolution was subjected to various tests to determine the difference in the results of the downsampling methods. Few criteria were implemented to check the similarity of the results. Since we are working with data amplification patterns in cancer, the first difference measure used is the number of amplifications produced by the three downsampling methods. Total number of differences in each chromosome band was computed and compared between three different downsampling methods. Figure 6 depicts that the results of the three different downsampling process did not show significant differences with respect to the number of amplifications.

Scrutinizing the results further mean difference between the number of differences produced in the number of amplifications by the three methods in various chromosome bands was computed.

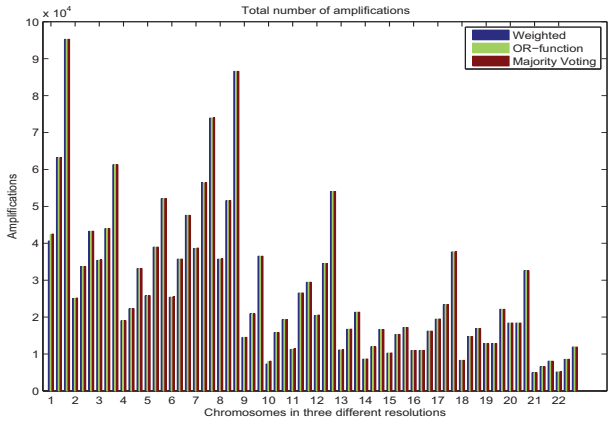


Figure 6: Total number of amplifications produced by the three different downsampling methods.

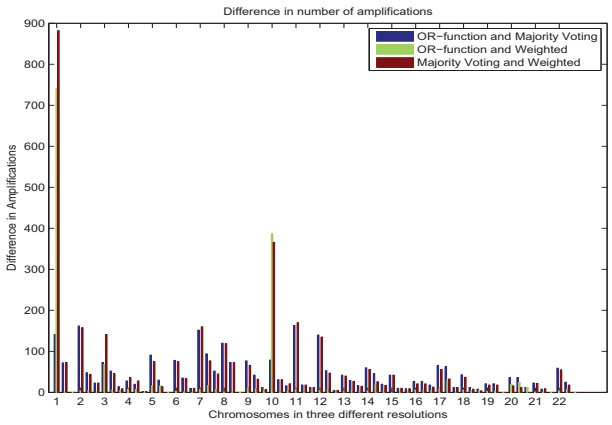


Figure 7: Difference in total number of amplifications produced by the three different downsampling methods.

Figure 7 suggests that there are differences in the results produced by the three downsampling methods with respect to the downsampling methods. However, the difference between the methods are not significant when the number of amplifications are considered. Similarly, other trivial difference measures such as row and column margins, and number of unique rows were also studied and the results showed that results of the downsampling methods are fairly similar.

However, these trivial measures used to calculate the difference are susceptible to some errors where the number of amplifications are same and also the number of amplifications does not change in different rows. For example, these methods does not show difference between the following two datasets.

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

In order to capture these differences, we further analyzed the difference between the two methods as the difference between the two resulting matrices for different methods using

standard matrix difference measures. The distance measure used is the square of the Frobenius norm [29] between two matrices. In binary matrices, Frobenius norm is essentially the number of cells where the two matrices differ.

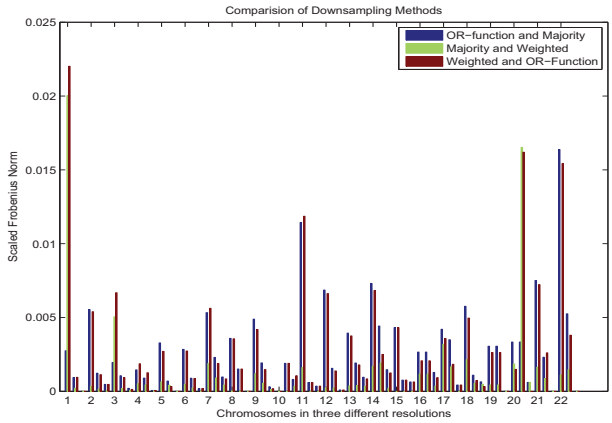


Figure 8: Comparison of three different downsampling methods : The difference measure used is scaled Frobenius norm.

Figure 8 suggests that the three downsampling methods produces fairly similar results. The Figure 8 also suggests that the differences are high in chromosome-1 which is expected because chromosome-1 is the largest chromosome. Differences are also high in lower resolution when compared to higher resolution because it is the lower resolution where the most changes takes place. The differences in the smaller chromosomes especially 20-22 are because of significant variation in the bands combined. Normally, three bands in finer resolution are combined in coarser resolution but in small chromosomes, the number of chromosome bands combined is very different thus making it difficult for weighted and OR-function downsampling method to work. It is to be noted that in the chromosomes where the differences are larger have larger number of differences in number of chromosome bands in different resolutions.

7.2 Model selection in Mixture Model

The size of the chromosome in terms of chromosome bands varied significantly. Some chromosomes had higher number of bands and some chromosomes had lower number of chromosome bands. Data from different resolutions were individually subjected to the mixture models. For model selection, for each mixture component, 50 models were trained using training set. It is often recommended to repeat cross-validation technique a number of times because 10-fold cross-validation can be seen as a "standard" measure of the performance whereas ten 10-fold cross-validations would be a "precise" measure of performance [30]. Since EM-algorithm is sensitive to the initializations and the results may differ on the same data for different initializations and it can get stuck in local minima and the global optimum results are not often guaranteed [31], 50 different models were trained for each number of components. In other words, 10-fold cross-validation was repeated 50 times. The number of mixture components was varied from 2 to 20 for all chromosomes in all resolutions. Validation set for each model is the one remaining subset of the data which is not used for training. Total likelihood for the training data as well as the validation data is calculated and averaged for each mixture component. The number of components for which the likelihood is maximum is selected as the model for the data taking parsimony into account. In other words, in some cases, models with lesser mixture components are selected instead of models with large number of mixture components for which likelihood was higher. Model selection was performed on all chromosomes as chromosome-wise analysis can reveal interesting facts about amplification of specific chromosomes and guarantees efficient computation & ease of analysis. Here, results are explained only for chromosome-17 as an example. Two sets of original data were available in resolution 393 and 850. Experiments were performed in the original resolution and sampling was performed to sample the data to different resolutions. Experiments showed that number of components required to optimally fit the model is independent of the resolution of the data thus showing that the significant patterns are not lost during sampling. Figure 9 and 10 show a model selection procedure for the data in resolution 393 and 850.

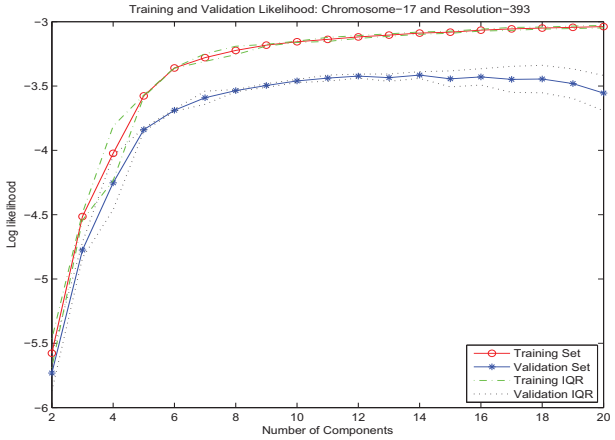


Figure 9: Model selection for the the original data in resolution 393. The averaged log-likelihood for training and validation sets in a 10-fold cross-validation setting for different number of components in chromosome-17: Resolution-393. The interquartile range(IQR) for 50 different training and validation runs have also been plotted.

Figure 9 shows the model selection in case of resolution 393 which downsampled from resolution 850. Figure 9 shows that the likelihood is smoothly increasing function with respect to the number of components. From Figure 9, it can be seen that validation likelihood is maximum when the number of components is 14, but instead of 14 components, 6 components was selected. It is to be noted that sometimes complex models overfit the data. Simple model also reduces the time and space complexity. Furthermore, the training and validation likelihood when the number of components is 6 are -3.3593 and -3.6883. In addition, when the number of components is 14, the training and validation likelihood are -3.0887 and -3.4146. Hence the difference in likelihood is negligible when compared with the efficiency in terms of time and space complexity. Furthermore, when the number of components are increased, IQR shows significant varia-

tion. The variation in IQR is because when the number of components are increased, samples can be assigned to different clusters. Additionally, the data in resolution in 393 was upsampled to resolution 850 and similar approach for model selection was followed.

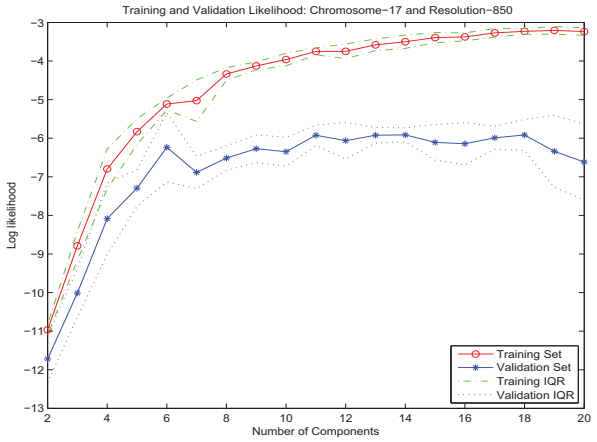


Figure 10: The averaged log-likelihood for training and validation sets in a 10-fold Cross validation setting for different number of components in chromosome17: Resolution 850. The interquartile range(IQR) for 10 different training and validation runs have also been plotted.

Figure 10 also shows that the IQR varies significantly from the mean. The choice of the number of components is straightforward because Figure 10 clearly shows a maximum of validation likelihood when the number of components is 8. Even when the number of components is 8, the variation in IQR is high. The variation in IQR can be compensated with sufficient training and would produce favorable results. The results can be further improved when the size of the dataset is increased.

The major aim of upsampling and downsampling was to aid in the integration of databases. The clinical aspects regarding the classification of cancer with mixture models is already established in [11] and [12]. Thus, data in different resolution were combined after upsampling and downsampling and model selection was performed. Table 2 summarizes the results of the experiments on chromosome: 17. To calculate the Likelihood 50 different models were trained to convergence and likelihood of the data was calculated for each model and the mean of the results are reported.

Data Resolution	J	Likelihood
Original in 393	8	-3.39
Original in 850	8	-4.75
Downsampled to 393	6	-3.41
Upsampled to 850	6	-5.23
Combined in 393	7	-3.36
Combined in 850	7	-5.11

Table 2: Results of experiments on chromosome-17. J denotes the selected number of component distributions.

Table 2 shows the number of components required to fit the data differs in different resolution and different number of samples in the data. The likelihood of data in higher resolution is lower than the likelihood of the data in the lower resolution when the number of components are same. This phenomenon can be attributed to the curse of dimensionality [32]. For example, the dimensionality of data in resolution 393 and 850 differs by 12 in chromosome-17 but likelihood is lesser even when the number of components is similar. For the original data in resolution 393 and 850, the difference in number of parameters of the model is $6 * (1 + 26) - 6 * (1 + 18) = 48$ which invites significant amount of computational complexity. The increased complexity however does not produce corresponding the increase in the likelihood. With increasing samples, the number of components are not increased because the complexity of mixture models depends on the complexity of the problem being solved, not with the size of dataset. This experiments with the mixture models also shows that patterns present in the higher resolution of the data is efficiently and effectively preserved in lower resolution.

Data Resolution	# X	Train	Test
Original in 393	342	0.25	0.06
Original in 850	2716	0.43	0.30
Downsampled to 393	2716	1.12	0.20
Upsampled to 850	342	2.16	0.08
Combined in 393	3058	1.43	0.19
Combined in 850	3058	2.51	0.32

Table 3: Computational complexity for training and testing of a single mixture model with appropriate number of mixture components as decided in 2. Experiments are performed on chromosome-17 and time is calculated in seconds. X denotes the number of data samples. The hardware used is Intel Core2Duo 2.00GHz CPU with a memory of 3 GB.

The major drawback in using mixture models is computational complexity of training the mixture models. Normally, training mixture models are computationally expensive when compared to other parametric (such as Poisson distribution) as well as non-parametric (such as k-means) methods. Similar to other machine learning methods computational complexity of the mixture model also increases with increasing dimension i.e. resolution in our case. Thus, computational complexity was also estimated for each resolution for the number of components shown in Table 2. As shown in the Table 3 the computational complexity increases with increasing resolution. To estimate the training time, 50 different models are trained until 10 iterations and the mean of the result is taken as final training time. Similarly, likelihood is calculated for 50 different models trained to calculate the training time and the mean of the results is reported. Experiments with resolution 850 required approximately twice the time required for the resolution 393. Furthermore, from Table 2, we also know that number of components required is high when the resolution is increased but the likelihood decreases. In addition, the curves are smoother in Figure 9 when compared to Figure 10. This phenomenon is because of the intrinsic problems of working with high dimensional data arising in higher resolution. These results suggest that data in lower resolution is preferred but lower resolution does not capture all the available biological information. Thus, there is a trade-off between the two.

7.3 Frequent itemsets

The measure of frequent itemsets provides a metric for the similarity measure between the sampled data and original data. Furthermore, our major aim was to upsample and downsample the data so that the patterns in the original resolution were retained. Mining maximal frequent itemset in the context of mixture modelling of multivariate Bernoulli distribution is two fold. It has by shown in [14] that maximal frequent itemset can be used to describe the finite mixture of multivariate Bernoulli distributions compactly and in a language understandable by the domain experts. In [14], the authors implemented a mixture of Bernoulli distributions in clustering binary data to derive frequent itemsets from the cluster-specific data sets and found that the cluster-specific maximal frequent itemset were significantly different from those itemsets extracted globally.

Similar to [14], we used MAFIA (MAximal Frequent Itemset Algorithm) [8] to mine the frequent patterns because other similar algorithms such as Apriori [6] would produce long results which will be difficult to interpret. The frequency or the threshold was choosen as 0.5 motivated by a majority voting protocol. Upscaling is simple and is always guaranteed to retain the frequent itemset although the number of frequent itemset increases with the exact same support. Therefore, they have not been reported.

Data	Maximal frequent itemsets
Og. 393	{11},{12}
Og. 850	{7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24}
OR. 393	{5, 6, 7, 8, 9, 10, 11, 12}
We. 393	{7,8}, {5, 6, 7}, {7,12}, {7,11}, {8, 9, 10, 11, 12}
Mj. 393	{5, 6, 7, 8, 9, 10, 11, 12 }
Co. 393	{5, 6, 7}, {6,7,8}, {7, 8, 9, 10, 11}, {7, 8, 11, 12}, {8, 9, 10, 11, 12}
Co. 850	{7, 8, 9}, {8, 9, 10, 11, 12, 13, 14}, {9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21}, {9, 10, 11, 12, 13, 14, 19, 20, 21, 22, 23, 24 }, { 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24}

Table 4: Maximal frequent itemsets for data in different resolutions. The threshold used is 0.5. Og., OR., We., Mj. and Co. denotes the original, OR-function down-sampled, weighted down-sampled, majority voting downsampled and combined data respectively.

From Table 4, we can see that the maximal frequent itemsets are preserved during sampling of resolutions. For example, in OR-function downsampled data in resolution 393 and original data in resolution 850, there is no difference in the maximal frequent itemset because from upsampling Table 1 we know that items 7,8, and 9 in 850 represents items 5, 6 and 7. Items 8 to 14 in 850 are combined to form item 8 in the data. Other itemsets are also formed with similar combinations. Weighted downsampling differs more than other two types of methods but the difference is not significant. The results of sampling can be seen more profoundly in integrated datasets where each itemsets in higher resoulution can be defined by the frequent itemsets lower resolution. The differences in some cases are only seen because support for those itemsets are less; these differences can be expected because data in lower resolution can not encompass all the information in higher resolution.

8. SUMMARY AND CONCLUSIONS

A simple upsampling and three different downsampling methods were proposed and their results were studied. The results were plausible and fairly consistent. The resulting data in different resolutions efficiently captures the information of data in different resolutions. Mixture models were then applied to the data in different resolutions. Finally, data in two different resolutions were integrated and then analyzed in one resolution. The results suggested that number of components required to fit the data does not differ across resolutions but likelihood of the model on higher resolution is poor than on lower resolution although the data is the same but representation is different. The clustering results of mixture models possesses high clinical significance. Furthermore, the maximal frequent itemsets and mixture modelling show that significant patterns in the data is maintained during sampling.

9. FUTURE WORK

Mixture models are limited because they work with one resolution of data. In the future work, they can be extended to work with multiple resolutions of the data where the sampling is incorporated with in models. The sampling techniques can be constrained to maintain the significant patterns in the dataset.

10. REFERENCES

[1] L.G. Shaffer and N. Tommerup. *ISCN 2005: An International System for Human Cytogenetic Nomenclature(2005) Recommendations of the International Standing Committee on Human Cytogenetic Nomenclature*. Karger, 2005.

[2] A. Kallioniemi, O.P. Kallioniemi, D. Sudar, D. Rutovitz, J.W. Gray, F. Waldman, and D. Pinkel. Comparative genomic hybridization for molecular cytogenetic analysis of solid tumors. *SCIENCE*, 258(5083):818–821, OCT 30 1992.

[3] D. Pinkel, R. Segraves, D. Sudar, S. Clark, I. Poole, D. Kowbel, C. Collins, W.L. Kuo, C. Chen, Y. Zhai, S. H. Dairkee, B.M. Ljung, J.W. Gray, and D.G. Albertson. High resolution analysis of DNA copy number variation using comparative genomic hybridization to microarrays. *Nature Genetics*, 20: 207 – 211, 1998.

[4] I. K. Fodor. A survey of dimension reduction techniques. Technical report, U.S. Department of Energy, June 2002.

[5] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, New York, NY, USA, 1993. ACM.

[6] H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *AAAI Workshop on Knowledge Discovery in*

Databases (KDD-94), pages 181–192, Seattle, Washington, 1994. AAAI Press.

[7] Arianna Gallo, Pauli Miettinen, and Heikki Mannila. Finding subgroups having several descriptions: Algorithms for redescription mining. In *SDM*, pages 334–345, 2008.

[8] Doug Burdick, Manuel Calimlim, and Johannes Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *In ICDE*, pages 443–452, 2001.

[9] J.R. Pollack, C.M. Perou, A.A. Alizadeh, M.B. Eisen, A. Pergamenschikov, C.F. Williams, S.S. Jeffrey, D. Botstein, and P.O. Brown. Genome-wide analysis of dna copy-number changes using cdna microarrays. *Nature Genetics*, 23(1):41–46, 1999.

[10] S. Knuutila, Y. Aalto, K. Autio, A. Björkqvist, W. El-Rifai, Hemmer S., T. Huhta, E. Kettunen, S. Kiuru-Kuhlefelt, M.L. Larramendy, T Lushnikova, O. Monni, H. Pere, J. Tapper, M. Tarkkanen, A. Varis, V. Wasenius, M. Wolf, and Y. Zhu. Dna copy number losses in human neoplasms. *Gynecologic Oncology*, 155(2):683–694, 1999.

[11] S. Myllykangas, J. Himberg, T. Böhling, B. Nagy, J. Hollmén, and S. Knuutila. DNA copy number amplification profiling of human neoplasms. *Oncogene*, 25(55):7324–7332, 2006.

[12] S. Myllykangas, J. Tikka, T. Böhling, S. Knuutila, and J. Hollmén. Classification of human cancers based on DNA copy number amplification modeling. *BMC Medical Genomics*, 1:15, 2008.

[13] J. Tikka, J. Hollmén, and S. Myllykangas. Mixture modeling of DNA copy number amplification patterns in cancer. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4507 LNCS:972–979, 2007.

[14] J. Hollmén and J. Tikka. Compact and understandable descriptions of mixtures of bernoulli distributions. *Lecture Notes in Computer Science including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*, 4723 LNCS:1–12, 2007.

[15] P.M.V. Rancoita, M. Hutter, F. Bertoni, and I. Kwee. Bayesian DNA copy number analysis. *BMC Bioinformatics*, 10, 2009.

[16] B. D'haene, J. Vandesompele, and J. Hellemans. Accurate and objective copy number profiling using real-time quantitative PCR. *Methods*, 50(4):262–270, 2010.

[17] E. Despierre, D. Lambrechts, P. Neven, F. Amant, S. Lambrechts, and I. Vergote. The molecular genetic basis of ovarian cancer and its roadmap towards a better treatment. *Gynecologic Oncology*, 117(2):358–365, 2010.

[18] L. Wall. Perl: Practical Extraction and Report Language. Website, 1987. http://www.perl.org/: Last Accessed: 15 Mar 2010.

[19] National Center for Biotechnology Information. Human genome project. Website, February 2010. `http://www.ncbi.nlm.nih.gov/projects/mapview/` Last Accessed: 5 Feb 2010.

[20] G. J. McLachlan and D. Peel. *Finite mixture models*, volume 299 of *Probability and Statistics – Applied Probability and Statistics Section*. Wiley, New York, 2000.

[21] B. S. Everitt and D. J. Hand. *Finite mixture distributions*. Chapman and Hall, 1981.

[22] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st ed. 2006. corr. 2nd printing edition, October 2007.

[23] S. Geisser. A predictive approach to the random effect model. *Biometrika*, 61(1):101–107, 1974.

[24] F. Monsteller and J. Tukey. Data analysis including statistics. *In Lindzey G. and Aronson E., editors, Handbook of Social Psychology, Vol-2, Addison-Wesley*, 1968.

[25] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal Of The Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[26] J. H. Wolfe. Pattern clustering by multivariate mixture analysis. *Multivariate Behavioral Research*, 5:329–350, 1970.

[27] J. Hollmén. *BernoulliMix: Program package for finite mixture models of multivariate Bernoulli distributions*, May 2009. Freely available in http://www.cis.hut.fi/jHollmen/BernoulliMix/.

[28] Mathworks. Matlab: the language of technical computing. Website, 1994. http://www.mathworks.com/products/matlab/: Last Accessed: 15 Mar 2010.

[29] G. W. Stewart. *Matrix Algorithms: Volume 1, Basic Decompositions*. Society for Industrial Mathematics, 1998.

[30] S.D. Gay. *Datamining in proteomics: extracting knowledge from peptide mass fingerprinting spectra*. PhD thesis, University of Geneva, Geneva, 2002.

[31] G. J. Mclachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley-Interscience, 1 edition, November 1996.

[32] W. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley, 2007.

CloseViz: Visualizing Useful Patterns

Christopher L. Carmichael
Department of Computer Science
The University of Manitoba
Winnipeg, MB, Canada
umcarmi1@cs.umanitoba.ca

Carson Kai-Sang Leung*
Department of Computer Science
The University of Manitoba
Winnipeg, MB, Canada
kleung@cs.umanitoba.ca

ABSTRACT

Numerous algorithms have been proposed since the introduction of the research problem of frequent pattern mining. Such a research problem has played an essential role in many knowledge discovery and data mining (KDD) tasks. Most of the proposed frequent pattern mining algorithms return the mined results in the form of textual lists that contain frequent patterns showing those frequently occurring sets of items. As "a picture is worth a thousand words", the use of visual representation can enhance the user understanding of the inherent relations in a collection of frequent patterns. Although a few visualizers have been developed to visualize the raw data or the results for some data mining tasks, most of these visualizers were not designed for visualizing frequent patterns. For those that were, they show all the frequent patterns that can be mined from datasets. It is not uncommon that, for many real-life applications, the user may end up be overwhelmed by such a huge number of patterns. In this paper, we propose a visualizer—called *CloseViz*—to show the user only the *useful patterns*. Specifically, CloseViz shows only *closed frequent patterns*. By doing so, CloseViz reduces the number of displayed patterns to a useful amount while retaining all the important frequency information. Moreover, CloseViz presents the closed frequent patterns to the user in a useful manner, which allows visual exploration of the patterns. Note that the closed patterns shown by CloseViz can be considered as surrogates for all the frequent patterns that can be mined from the datasets.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*; H.1.2 [**Models and Principles**]: User/Machine Systems—*human factors*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

*Corresponding author: C.K.-S. Leung.

UP'10, July 25, 2010, Washington, DC, USA.

General Terms

Algorithms; Design; Experimentation; Human factors; Management; Measurement; Performance; Reliability

Keywords

Knowledge discovery and data mining (KDD), useful patterns, closed itemsets, frequent itemsets, reduction in the number of returned patterns, removal of redundant patterns, visual exploration of patterns

1. INTRODUCTION

Frequent pattern mining [1, 21] aims to discover implicit, previously unknown, and potentially useful knowledge in the form of *frequent patterns* (i.e., frequently occurring sets of items, which are also known as *frequent itemsets*) from datasets. Some examples of frequent patterns are sets of frequently purchased merchandise items, combinations of popular courses taken by students, groups of frequently observed features associated with edible mushrooms (or other kinds of food), common patterns in gene expression, and collections of web pages frequently visited by surfers. In many real-life situations, the mined frequent patterns can answer crucial questions that help users make important admininstrative and/or business decisions. The following are some examples of practical applications in real-life situations:

- Owners of an electronic store may want to find out how frequently certain kinds of items (e.g., audio cables, batteries) are purchased individually and how frequently they are purchased together? What kinds of items are frequently purchased together with DVD players (e.g., {audio cables, batteries, computers, DVD players})? Answers to these questions help the owners in item shelving and inventory.

- University administrators may want to know which popular courses (e.g., {Astronomy 101, Biochemistry 101, Computer Science 101}) are frequently taken together by students? Knowing this information may help the administrators in course scheduling.

In addition, frequent pattern mining often plays an essential role in various knowledge discovery and data mining (KDD) tasks such as the mining of association rules, sequences, episodes, and constrained patterns. This explains why, over the past two decades, numerous frequent pattern mining algorithms [9, 19, 36, 37] have been proposed to help answer real-life questions including the above two. Most of

these algorithms focus on how to compute the frequent patterns as efficiently as possible. For instance, some of the algorithms are Apriori-based [1], while some of them are FP-tree based [11, 30]; some other algorithms enhance the mining performance by using techniques like hashing and segmentation [27]; and, some algorithms deal with incremental updating [26]. In general, most of the frequent pattern mining algorithms return a collection of frequent patterns in *textual form* (e.g., a very long unsorted list of frequent patterns). Consequently, users may not easily comprehend the discovered knowledge and useful information from this lengthy list.

Presenting a collection of frequent patterns in *graphical form* can show the relations embedded in the data and help users understand the nature of the useful information and discovered knowledge. Hence, researchers have also considered visual analytics [17, 20, 34, 38, 40] and visualization techniques [8, 13, 15] to assist users in gaining insight into massive amounts of data or information. Several visualization systems [2, 16, 35] have been developed for visualizing data (i.e., input of the KDD process). Other systems have also been developed to visualize the mining results (i.e., output of the KDD process) such as clusters [18], decision trees [3], or association rules [5, 12]. However, *not* many systems were designed to visual frequent patterns.

Recently, some visualizers have been designed for showing frequent patterns. For example, Yang [39] developed a system that can visualize frequent patterns. However, his system was primarily designed to visualize association rules, and it does not scale very well in assisting users to immediately see certain useful information (such as exact frequencies or support) of a huge number of frequent patterns. Munzner et al. [28] presented a visualizer called PowerSetViewer (PSV), which provides users with guaranteed visibility of frequent patterns in the sense that the pixel representing a frequent pattern is guaranteed to be visible by highlighting such a pixel. However, multiple frequent patterns may be represented by the same pixel. We previously proposed a visualization system—called FIsViz [24]—that aims to visualize frequent patterns. FIsViz represents each frequent pattern by a polyline in a two-dimensional space. The location of the polyline indicates the exact frequency of the pattern explicitly. As a result, FIsViz enables users to visualize the mined results (i.e., frequent patterns) for many practical real-life applications. However, in some other applications (especially, when the number of frequent patterns is huge), FIsViz may not scale very well. Users may require more effort to be able to clearly visualize frequent patterns. The problem is caused by the use of polylines for representing frequent patterns. As the polylines can be bent and/or can cross over each other, it can be difficult to distinguish one polyline (representing a frequent pattern) from another.

To avoid the bending and crossing-over of polylines, we subsequently proposed (i) another visualizer called WiFIsViz [25] and (ii) a visualization module called FpViz [23] in a visual analytic tool called FpVAT [22]. Both WiFIsViz and FpViz represent each frequent pattern by a horizontal line in a two-dimensional space. The location of the horizontal line indicates the exact frequency of the pattern explicitly. Hence, they enable users to visualize the mined results. However, in some real-life applications, the number of patterns returned by frequent pattern mining algorithms can be so huge that *not* all of the patterns can be clearly visualized and displayed on a single screen.

In this paper, we investigate how to reduce the number of patterns returned by frequent pattern mining algorithms and/or displayed by frequent pattern visualizers? We also investigate how to do so while retaining the important information (e.g., frequency or support value) of the patterns? The **key contribution** of our work in this paper is the proposal and development of a novel interactive and scalable *useful pattern visualizer*, called *CloseViz*, which shows only closed frequent patterns. Note that closed frequent patterns are condensed and concise representation of all frequent patterns. By showing only closed patterns, CloseViz greatly reduces the number of useful patterns to be shown or returned to users. It also provides users with effective visual support in the data analysis and KDD process for visual exploration of the patterns. As CloseViz retains the frequency information of patterns, the displayed closed patterns mined from a dataset can be considered as surrogates for all the frequent patterns that can be mined from the same dataset.

This paper is organized as follows. Next section briefly describes related work. In Section 3, we introduce our CloseViz and describe its design. In Section 4, we present interactive features of CloseViz, which allows users to visually explore useful patterns. Section 5 shows evaluation results. Finally, conclusions, as well as ongoing and future work, are presented in Section 6.

2. RELATED WORK

In the field of KDD, several effective systems have been developed for visualizing raw data. Examples include Spotfire [2], independence diagrams [4], VisDB [16], and Polaris [35]. Besides them, there are also systems that focus on visualizing the results of KDD tasks other than frequent pattern mining (e.g., clustering [18, 33], classification [3, 10], and association rule mining [5, 6, 12]). In addition, there were also systems that can be used for visualizing frequent patterns [24, 25, 28, 39, 41]. We briefly discuss some of them in the remainder of this section.

2.1 Yang's system

Yang [39] designed a system mainly to visualize association rules—but can also be used to visualize frequent patterns—in a two-dimensional space consisting of many vertical axes. In his system, all domain items are sorted according to their frequencies and are evenly distributed along each vertical axis. A frequent pattern consisting of k items (i.e., a k-itemset) is then represented by a curve that extends from one vertical axis to another connecting k such axes. The thickness of the curve indicates the frequency (or support) of such a frequent pattern. However, such a representation suffers from a few problems. First, the use of thickness only shows *relative* (but not *exact*) frequencies of the patterns. Comparing the thickness of curves is not easy. Second, since items are sorted and *evenly* distributed along the axes, users only know some items are more frequent than the others, but cannot get a sense of how these items are related to each other in terms of their exact frequencies (e.g., whether item a is twice as frequent as, or just slightly more frequent

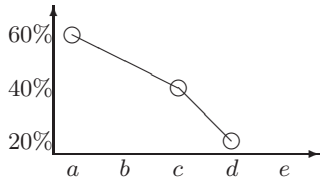


Figure 1: FIsViz [24] shows $\{a, c, d\}$.

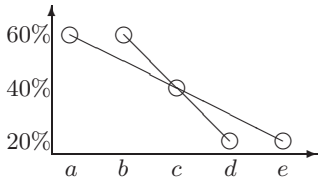


Figure 2: FIsViz [24] shows (i) $\{a, c, d\}$ & $\{b, c, e\}$ or (ii) $\{a, c, e\}$ & $\{b, c, d\}$.

than, item b). As a preview, our proposed CloseViz provides users with *exact* frequency information instead.

2.2 PowerSetViewer

Unlike Yang's system, PowerSetViewer (PSV) [28] was designed specifically to visualize frequent patterns in the context of the powerset universe. With PSV, all patterns are grouped together based on cardinality and are presented in a two-dimensional grid. A different background colour is assigned to each cardinality, and patterns of the same cardinality are mapped into consecutive grid squares. When the number of patterns exceeds the number of allocated grid squares, PSV maps several patterns into the same grid square. A square is highlighted to indicate that it contains at least one frequent pattern. This provides users with guaranteed visibility. However, PSV also suffers from a few problems. First, as a highlighted grid square may contain multiple frequent patterns, it is not easy to tell distinguish one pattern from all that mapped into the same square. Second, like Yang's system, PSV also does *not* show the exact frequency of a pattern.

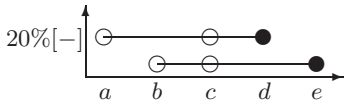
2.3 FIsViz

Frequent itemset visualizer (FIsViz) [24] is one of the recently developed visualizers that were designed to show frequent patterns. It represents a frequent pattern consisting of k items (i.e., k-itemset) by a polyline that connects k nodes (where each node represents an item in the k-itemset) in a two-dimensional space. The frequency of the i-th prefix of a pattern X is indicated by the y-position of the i-th node in the polyline representing X. For example, when $X = \{a, c, d\}$ as shown in Figure 1, the frequencies of its prefixes $\{a\}$ and $\{a, c\}$ (i.e., 60% and 40%) are respectively indicated by the y-positions of nodes a and c in the polyline. Similarly, the frequency of $X = \{a, c, d\}$ (i.e., 20%) is represented by the y-position of node d in that polyline.

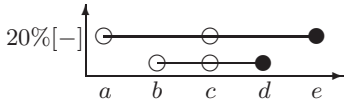
With this representation, slopes of different sectors of a polyline can vary. In other words, the entire polyline may not be a straight one (i.e., it may be bent). Moreover, polylines representing different patterns may cross each other. This may make it difficult for users to distinguish one sector of a polyline from another. For example, is Figure 2 showing (i) $\{a, c, d\}$&$\{b, c, e\}$ or (ii) $\{a, c, e\}$&$\{b, c, d\}$? As a preview, our proposed CloseViz avoids the crossing-over of polylines; instead, it uses horizontal lines to represent frequent patterns.

2.4 WiFIsViz and FpViz

WiFIsViz [25] and FpViz [23] are two other visualizers that were designed for visualizing frequent patterns. The key difference between the two is that the former uses two half-screens to visualize the frequent patterns (a half-screen displays all the frequent patterns and the other half-screen shows their frequencies) and the latter shows all the frequent patterns and their frequencies on the same full-screen. Like FIsViz, both WiFIsViz and FpViz show a pattern consisting of k items (i.e., k-itemset) in a two-dimensional space. Unlike FIsViz, they use an orthogonally laid out node-link diagram instead of the polyline diagram. Specifically, they represent the k-itemset by a horizontal line connecting k circles (where each circle represents an item in the k-itemset). By doing so, they reduce the number of line crossings, which in turn improves the legibility of the display. They also minimize bends (as bends occur only at 0° or 90° angles), which further enhances the legibility of the display. Recall from Figure 2 that, when using FIsViz, users may have difficulties in distinguishing (i) the pair $\{a, c, d\}$ & $\{b, c, e\}$ from (ii) the pair $\{a, c, e\}$ & $\{b, c, d\}$. In contrast, WiFIsViz and FpViz show (i) frequent patterns $\{a, c, d\}$ & $\{b, c, e\}$ as follows:



and (ii) frequent patterns $\{a, c, e\}$ & $\{b, c, d\}$ as follows:



Hence, with WiFIsViz and FpViz, users can easily distinguish between these two pairs of frequent patterns.

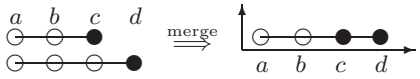
While these two visualizers effectively display all the frequent patterns that can be mined from a dataset, they may suffer from the following potential problems.

Potential Problem 1. Both WiFIsViz and FpViz show *all* the frequent patterns. In very large datasets for many real-life applications, it is not unusual that a huge number of frequent patterns can be returned due to pattern exploration. Note that, given a domain of m items, there are potentially $2^m - 1$ non-empty frequent patterns.

Potential Problem 2. The two visualizers use different types of icons (e.g., filled circles, unfilled circles) for representing items within a pattern. As such, users may not fully understand or remember the meanings of filled and unfilled circles. In very large datasets, it is not unusual for multiple frequent patterns to have the same frequency. To reduce the number of horizontal lines representing frequent patterns of the same frequency and to attempt to squeeze all horizontal lines onto the screen display, both WiFIsViz and FpViz apply several compression techniques. One of them is to merge two frequent patterns X and Y of the same frequency if X is a prefix of Y. Then, to distinguish X from Y in the merged

result, the visualizers fill the circles corresponding to the last items of X and Y. See Example 1.

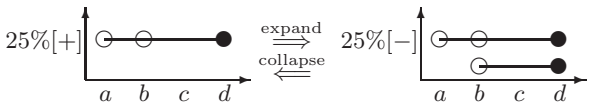
Example 1. Given frequent patterns $\{a, b, c\}$ & $\{a, b, c, d\}$ of the same frequency, both WiFIsViz and FpViz merge their corresponding lines into one line connecting four circles representing items a, b, c & d in the patterns. To indicate that $\{a, b, c\}$ is a prefix of $\{a, b, c, d\}$, the visualizers fill the circle c representing the last item of $\{a, b, c\}$ and the circle d representing the last item of $\{a, b, c, d\}$ as shown below:



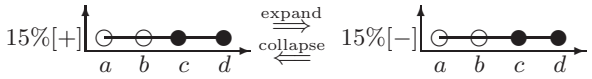
However, without prior training and/or reminder, users may not know or may forget the meaning of the filled circles. Our very recent evaluation showed that, in these situations, users sometimes wondered why some circles were filled and some were not. □

Potential Problem 3. With WiFIsViz and FpViz, users may perform some unnecessary actions. Recall that merging multiple horizontal lines that represent a frequent pattern and its prefixes (e.g., lines representing $\{a, b, c, d\}$ and its prefix $\{a, b, c\}$ were merged into a single line in Example 1) is one of the compression techniques. Another compression technique is to collapse several horizontal lines representing frequent patterns of the same frequency onto a single horizontal line. In order to know the details of the patterns embedded within the resulting line, the two visualizers provide users with a [+] button for expanding such a line. In general, the visualizers provide users with a [+] button for every line. As some of these lines may not be the result of collapsing multiple horizontal lines. Consequently, expanding these lines is unnecessary because these lines are identical to their expansion. See Example 2.

Example 2. Consider two frequent patterns $\{a, b, d\}$ and $\{b, d\}$ of the same frequency (say, 25%). When using this compression technique, the two horizontal lines are collapsed onto one connecting three circles (representing items a, b and d) with the circle for d filled. Users need to expand such a line by clicking the [+] button in order to see that such a "collapsed" line represents two frequent patterns $\{a, b, d\}$ and $\{b, d\}$, as shown below:



In real-life applications, many lines may need to expand. However, at the same time, there may exist lines that do not need to expand. For instance, reconsider the two frequent patterns $\{a, b, c\}$ and $\{a, b, c, d\}$ in Example 1. If they are the only two patterns of a particular frequency (say, 15%), then it is unnecessary to expand the line for frequency 15%. The reason is that, as one frequent pattern is a prefix of another, the resulting line is just a result of merging the prefix with its "extension" (i.e., *not* a result of collapsing multiple lines). Hence, the expanded view is identical to its collapsed view, shown as follow:



Therefore, it is an unnecessary action to expand such a line. □

To cap, while WiFIsViz and FpViz reduce the number of bends and crossover of polylines by using horizontal lines, they both may suffer from a few potential problems when handling very large datasets. First, the number of patterns to be displayed can be huge, especially when using very low minimum support threshold *minsup*. Thus, showing only some patterns may help. Second, horizontal lines often connect two types of circles (filled and unfilled ones). Users may not know or may forget the reason why some circles are filled and some are not. Third, there are still a lot of horizontal lines that can be expanded as every line is accompanied with a [+] button for potential expansion. However, some of these lines do not need to be expanded as their expanded views would be identical to their collapsed views.

3. CloseViz: VISUALIZING CLOSED FREQUENT PATTERNS

In this section, we propose a simple yet powerful visualizer, called *CloseViz*, for showing useful patterns in the form of closed frequent patterns.

3.1 Showing Only Closed Patterns

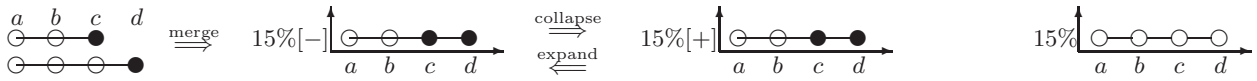
To provide users with useful patterns among the exponential number of frequent patterns and to avoid presenting redundant information to users, our proposed CloseViz only shows *closed frequent patterns* (instead of all the frequent patterns).

Definition 1. (Closed Pattern [29, 42]). A pattern X, which is a non-empty subset of all domain items, is *frequent* if its frequency (or support value) is no less than the user-specified minimum support threshold *minsup*. Then, a frequent pattern X is *closed* if there does not exist any proper superset of X having the same frequency as X. □

In other words, if there are two frequent patterns X and Y such that (i) Y is a proper subset of X and (ii) they both have the same frequency, then CloseViz does not display Y because Y is not a closed frequent. Since (i) X is a proper superset of Y and (ii) X has the same frequency as Y, it would be redundant to display Y. By visualizing only closed frequent patterns, CloseViz greatly reduces the number of useful patterns to be displayed. This solves Potential Problem 1.

It is important to note that, *while CloseViz reduces the number of patterns to be visualized, it retains all the frequency information* (cf. showing maximal patterns, which loses frequency information). Recall that a frequent pattern Z is *maximal* if no proper superset of Z is frequent.

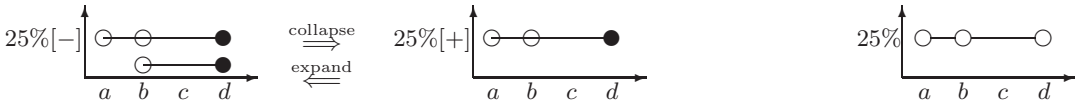
At the back end, our proposed CloseViz can be connected to any closed frequent pattern mining algorithm (e.g., A-Close [29], CLOSET [30], CHARM [43]), which mines closed frequent patterns from datasets. Once the mining algorithm found closed patterns, CloseViz effectively displays them so that users can explore the mined results visually. Alternatively, CloseViz can also be connected to any frequent



(a) FpViz [23] shows frequent patterns $\{a, b, c\}$ and $\{a, b, c, d\}$

(b) CloseViz shows the same patterns

Figure 3: Comparison of two visualizers in showing $\{a, b, c, d\}$ and its prefix $\{a, b, c\}$ (Example 3).



(a) FpViz [23] shows frequent patterns $\{a, b, d\}$ and $\{b, d\}$

(b) CloseViz shows the same patterns

Figure 4: Comparison of two visualizers in showing $\{a, b, d\}$ and $\{b, d\}$ (Example 4).

pattern mining algorithm (e.g., Apriori [1], FP-growth [11]), which finds all frequent patterns (including closed patterns as well as non-closed ones). With this alternative, once the mining algorithm found all frequent patterns, CloseViz performs an extra step to filter out all non-closed patterns from all the frequent patterns, and it then displays only the closed ones.

At the front end, CloseViz shows closed frequent patterns in a two-dimensional space. The x-axis shows the m domain items. CloseViz allows users to specify their preference on the ordering of these domain items (e.g., put those items of interest such as promotional items on the left and less interesting items on the right side of the x-axis). The default is lexicographical order. The y-axis shows the frequencies of the closed frequent patterns. Here, CloseViz provides users with two options:

1. *Use a linear scale to show all possible frequency values.* For instance, given three closed patterns of frequencies 20%, 40% and 80%, CloseViz with this option uses a linear scale so that users can easily get some insight about the relative frequencies of these closed patterns at a glance. Practically, users can easily notice that the closed pattern near the top portion of the display (at frequency 80%) is twice as frequent as the one slightly below the middle portion of the display (at frequency 40%), which in turn is twice as frequent as the one near the bottom portion of the display (at frequency 20%).

2. *Show only existing frequency values.* For instance, given three closed patterns of frequencies 20%, 40% and 80%, CloseViz with this option evenly divides the vertical space into three regions (one for each of the three frequency values 20%, 40% and 80%) so that no space is wasted for non-existing frequency values. As more space is allocated for existing frequency values, users can clearly visualize the useful patterns.

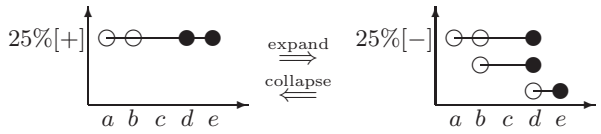
In general, CloseViz represents each closed pattern X consisting of k items (i.e., k-itemset) by a horizontal line connecting k circles, where each circle represents an item within X. The frequency of X is then represented by the y-position of the line representing X.

3.2 Representing Closed Patterns with One Type of Icons/Circles

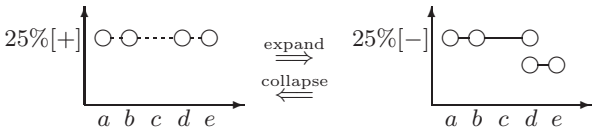
Even though the number of closed patterns is usually smaller than that of frequent patterns, it is not impossible for two closed patterns to have the same frequency. In these situations, CloseViz applies some compression techniques, which are similar—but not identical—to those of WiFIsViz or FpViz. Recall from Example 1 that, given frequent patterns $\{a, b, c\}$ and $\{a, b, c, d\}$ of the same frequency, WiFIsViz and FpViz merge the two horizontal lines corresponding to these two frequent patterns into a single line and fill some circles. The sole reason of using two different types of icons (namely, filled and unfilled circles) in WiFIsViz and FpViz is to enable users to visualize both a pattern and its prefix (i.e., filled circle indicates the last item of an itemset) in the merged line. In contrast, the representation provided by CloseViz is much simpler. No such merge is needed when CloseViz handles a pattern X with its prefix having the same frequency. The reason is that any prefix of X cannot be a closed pattern if X has the same frequency as its prefix. Hence, the prefix does not need to be shown. See Examples 3 and 4.

Example 3. Let us revisit Example 1, in which we are given two frequent patterns $\{a, b, c\}$ and $\{a, b, c, d\}$ of the same frequency (say, 15%). As shown in Figure 3(a), FpViz [23] merges the two horizontal lines corresponding to these two frequent patterns into a single line. To distinguish the prefix $\{a, b, c\}$ from $\{a, b, c, d\}$, FpViz fills the circles representing items c and d so as to indicate the last items of the two patterns. In contrast, our proposed CloseViz only needs to show $\{a, b, c, d\}$. The reason is that, for frequent pattern $\{a, b, c\}$, there exists a proper superset $\{a, b, c, d\}$ that has the same frequency as $\{a, b, c\}$. Thus, $\{a, b, c\}$ is *not* a closed frequent pattern. Consequently, CloseViz only shows $\{a, b, c, d\}$ as a horizontal line connecting four circles a, b, c and d (as shown in Figure 3(b)); it does not need to fill any of four circles. □

Example 4. Let us revisit Example 2, in which we are given two frequent patterns $\{a, b, d\}$ and $\{b, d\}$ both of frequency 25%. As one pattern is not a prefix of another, FpViz [23] does *not merge* the two horizontal lines corresponding to these two frequent patterns into a single line. Instead, FpViz uses another compression techniques: It *collapses* the two lines onto one so as to reduce the number of horizontal



(a) FpViz [23] shows $\{a, b, d\}$, $\{b, d\}$, and $\{d, e\}$



(b) CloseViz shows the same patterns

Figure 5: Comparison of two visualizers in showing $\{a, b, d\}$, $\{b, d\}$, and $\{d, e\}$ (Example 5).

lines to be displayed. Again, FpViz fills the circle representing item c so as to indicate the last items of the two patterns. See Figure 4(a). In contrast, our proposed CloseViz only needs to show $\{a, b, d\}$. The reason is that, for frequent pattern $\{b, d\}$, there exists a proper superset $\{a, b, d\}$ that has the same frequency as $\{a, b, d\}$. Thus, $\{b, d\}$ is *not* a closed frequent pattern. Consequently, CloseViz only shows $\{a, b, d\}$ as a horizontal line connecting three circles a, b and d (as shown in Figure 4(b)); it does not need to fill any of three circles. □

3.3 Expanding a Line in the Collapsed View Only when Needed

Our proposed CloseViz shows closed patterns normally in the (default) *collapsed view* so as to reduce the amount of vertical space required for displaying all closed patterns. As this collapsed view may hide some details, CloseViz provides users with an option to expand the "collapsed" line by clicking the [+] button. By so doing, users would be able to clearly obtain all the details. Note that the [+] button is not put for every line; it is put whenever it is needed. More specifically, CloseViz only puts a [+] button for a horizontal line when its collapsed view is different from its expanded view. In other words, if the collapsed view and the expanded view of a line are identical, CloseViz would not show the [+] button and thus users would not waste their time expanding a line that does not need expansion. Note that, besides clicking the [+] button to expand a "collapsed" line, users can also click the [−] button to collapse an expanded line. See Example 5.

Example 5. Let us revisit and modify Example 4. If we were to add a frequent pattern $\{d, e\}$ of frequency 25%, then FpViz [23] would collapse the three lines into a (*solid*) line connecting four circles with those for items d and e filled (as shown in Figure 5(a)). In contrast, our proposed CloseViz would show a *dashed* line connecting four circles and provide a [+] button. This would allow users to expand and reveal the details. When expanding such a line, CloseViz would show two lines. See Figure 5(b). □

Note that CloseViz uses dashed lines to represent the results of collapsing multiple lines onto one (cf. FpViz represents both the "collapsed" results as well as real frequent patterns using solid lines). Moreover, CloseViz only shows a [+] button for every dashed line, which requires expansion to reveal all the details; it does *not* show a [+] button for any solid line, which does *not* require expansion (cf. FpViz puts a [+] button on every line). Recall from Examples 3 and 4 that CloseViz does *not* put the [+] button for the line for closed pattern $\{a, b, c, d\}$ or $\{a, b, d\}$.

If two closed patterns happen to share a common prefix, CloseViz also collapses their corresponding lines onto a single line and provides a [+] button to allow users to expand or explore the details. See Example 6.

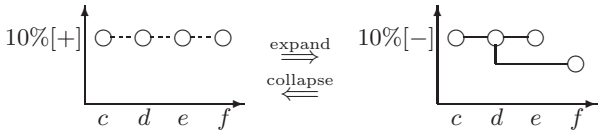


Figure 6: CloseViz shows $\{c, d, e\}$ and $\{c, d, f\}$ (Example 6).

Example 6. Given frequent patterns $\{c, d, e\}$ and $\{c, d, f\}$ of the same frequency, CloseViz collapses their horizontal lines onto one and provides users with the [+] button to expand the single line. Conversely, users can click the [−] button to collapse the expanded lines. See Figure 6. □

To recap, our proposed CloseViz solves Potential Problem 1 by showing only closed patterns instead of all frequent patterns. The number of closed patterns is usually much lower than the number of frequent patterns. Moreover, CloseViz solves Potential Problem 2 by using a much simpler representation (for closed patterns) that of FpViz (for all frequent patterns). CloseViz does not need to use both filled and unfilled circles; it only uses unfilled circles. Furthermore, CloseViz solves Potential Problem 3 by putting [+] buttons only for dashed lines, which represent results of collapsing multiple lines onto one. No [+] button is needed for solid lines, which represent real closed patterns.

3.4 Observations

The following are some observations on the representation of closed patterns and their frequencies when using CloseViz:

- If a closed pattern Y is a proper subset of another closed pattern X, then CloseViz must show Y at a frequency above X. The reason is that (i) the frequency of Y cannot be lower than that of X (due to the Apriori property of patterns) and (ii) the frequency of Y cannot be the same as that of X (due to the definition of closed patterns).
- CloseViz provides information about the presence of items in some patterns. For example, if a circle representing an item x appears on a *solid* horizontal line at a particular frequency (say, p%), then users can induce that x is contained in the closed pattern represented by such a solid line. On the other hand, if a circle representing an item x appears on a *dashed* horizontal line at frequency p%, then user can induce that x is contained in at least one closed pattern of frequency p%.
- Similarly, CloseViz provides information about the absence of items from any patterns. For example, if a circle representing an item x' does *not* appear on the horizontal line at a particular frequency (say, q%), users

can induce that x' is guaranteed not to appear in any closed pattern of frequency $q\%$.

- CloseViz greatly reduces the number of patterns to be shown by presenting only closed patterns, which are often a small subset of all frequent patterns.
- Most importantly, CloseViz *retains all frequency information* (i.e., no information loss). Users can easily deduce all frequent patterns and their frequencies based on the closed patterns (and their frequency information) presented by CloseViz.
- CloseViz only uses one type of icons—namely, unfilled circles—to present items within closed patterns. Thus, the presentation of information is clear. Users neither need to learn the meanings of filled and unfilled circles nor require distinguishing the filled circles from unfilled ones.
- CloseViz does *not* put [+] buttons on all lines. Buttons only appear on *dashed* lines because they are the results of collapsing lines that represent multiple closed patterns onto one line. To reveal details in the expanded view, users can click the [+] button and CloseViz expands the dashed line. Conversely, by clicking the [−] button on the expanded line, users get back the collapsed view.
- CloseViz does *not* put [+] on *solid* lines because they are *not* results of collapsing lines (which represent multiple closed patterns) onto one line. The expanded view and the collapsed view of these solid lines are the same. It would be a waste of resources and an unnecessary action to expand the solid lines.

4. SUPPORTING VISUAL EXPLORATION

In this section, we describe some interactive features of CloseViz, which enables users to perform visual exploration.

First, to help users to visually explore the mined (closed) patterns, our proposed CloseViz allows users to specify his preference on visualization of closed patterns. For example, if users are interested in finding those patterns containing some particular items (says, "computers"), users can click the x-labels of these items. CloseViz then ensures that all patterns containing these interesting items are clearly visible. To a further extent, as CloseViz does not fill any circle representing an item, circles could be filled with different colours to indicate different items of interest to users.

Second, CloseViz gives users an option to choose one or more relationships of some user-selected closed pattern X. Choices include (i) all closed proper prefixes of X, (ii) all closed proper subsets of X, (iii) all closed proper "extensions" of X, (iv) all closed proper supersets of X, etc. For example, given a closed pattern $X = \{b, c, d\}$ of frequency 36%, CloseViz may find the following closed patterns: Its proper prefixes $\{b\}$ and $\{b, c\}$ with frequency values 54% and 44% respectively, its proper subset $\{c, d\}$ of frequency 48%, its proper "extensions" $\{b, c, d, e\}$ and $\{b, c, d, e, f\}$ with frequency values 32% and 28% respectively, and its proper superset $\{a, b, c, d\}$ of frequency 24%.

Third, CloseViz permits users to select closed patterns of different cardinality to be displayed. By doing so, users do not need to view and explore all the closed patterns in one shot. Instead, users can view and explore some (e.g., closed 1-itemsets, closed 2-itemsets, etc.) at a time.

Fourth, CloseViz also provides users with details on demand (i.e., provides more details upon user request). In general, CloseViz gives users an overview of the entire collection of closed patterns and then allows users to interactively select parts of the patterns for which they request more details. For instance, when users hover the mouse over different parts of the display (say, hover on a sector of a dashed horizontal line), CloseViz shows a list of closed patterns represented by such a dashed line.

Fifth, CloseViz enables users to select a subset of domain items to be displayed on the x-axis and a smaller range of frequency values to be displayed on the y-axis. Once users made their selection, CloseViz only shows closed patterns involving the selected domain items and within the selected frequency range.

5. EVALUATION

In this section, we show our results on evaluating the proposed CloseViz.

5.1 Reduction in the Number of Displayed Useful Patterns

We compared our proposed CloseViz with some closely related visualizers (FIsViz [24], WiFIsViz [25], FpViz [23]). We used (i) several IBM synthetic datasets [1], (ii) some real-life databases (e.g., mushroom dataset) from UC Irvine Machine Learning Depository, and (iii) a student-course database for our university. The following were some observations:

- FIsViz presented frequent patterns as huge numbers of polylines, which were bent and crossed over each other. As such, it was not easy to effectively visualize the displayed frequent patterns.
- WiFIsViz avoided the bends and crossovers of polylines. However, it used a half-screen to show huge numbers of frequent patterns and another half-screen to show their frequencies. Since too many frequent patterns were squeezed onto a half-screen, it was not easy to effectively visualize the displayed frequent patterns.
- FpViz used a full-screen to display all frequent patterns. However, it used both filled and unfilled circles in the representation. Without a good understanding of the meaning of the filled vs. unfilled circles may make users puzzle about the interpretation of patterns. Moreover, all the horizontal lines were accompanied with [+] buttons so that users may end up expanding many lines that did not need expansion (e.g., the collapsed view of a horizontal line—which was not the result of collapsing multiple lines—was identical to its expanded view).
- CloseViz showed only closed patterns. For each of the datasets used in the evaluation, the number of closed patterns was much smaller than the number of all frequent patterns. Consequently, fewer lines were shown

by CloseViz. As they were horizontal lines, there were not crossovers of lines. Moreover, CloseViz only used one type of icons—namely, unfilled circles. As such, there was no need to distinguish filled circles from unfilled ones. Furthermore, CloseViz only put [+] buttons on dashed line (which needed expansion to reveal the details). In other words, when the line was not a result of collapsing multiple lines (i.e., a solid line, whose the collapsed view was identical to the expanded view), CloseViz did not put [+] buttons so that users did not waste their time in expanding a line for no extra information. See the following tables, which show some samples on the numbers of patterns returned by the above four visualizers and the numbers of lines displayed by these visualizers.

IBM synthetic dataset:

	#patterns	#lines
FIsViz	60,753	60,753 polyline sectors
WiFIsViz FpViz	frequent patterns	189 (expandible) horizontal lines
CloseViz	1,603 closed patterns	60 solid lines + 129 (expandible) dashed lines

Student-course database:

	#patterns	#lines
FIsViz	421	421 line sectors
WiFIsViz FpViz	frequent patterns	20 (expandible) horizontal lines
CloseViz	107 closed patterns	3 solid lines + 17 (expandible) dashed lines

As observed from the above tables, *CloseViz reduced the amount of information (#patterns) to be displayed* and the amount of work for exploration (#lines to be expanded).

5.2 Closed Patterns as Surrogates for Frequent Patterns

To assess the effectiveness of conveying closed patterns, we carried out a user study with CloseViz. The study was primarily case-based, within which two groups of users were required to answer different questions based on the visualizations of a given dataset (e.g., database containing information about courses taken by students). The scenario was that users need to identify closed patterns and make decisions based on their observations. We recruited 12 participants and separated them into two groups: (i) those who have data mining background and (ii) those who do not. None of the participants (regardless which of the two groups) was exposed to any form of visualization for frequent patterns—including our proposed CloseViz for visualizing closed frequent patterns. (As ongoing work, we are recruiting more participants.)

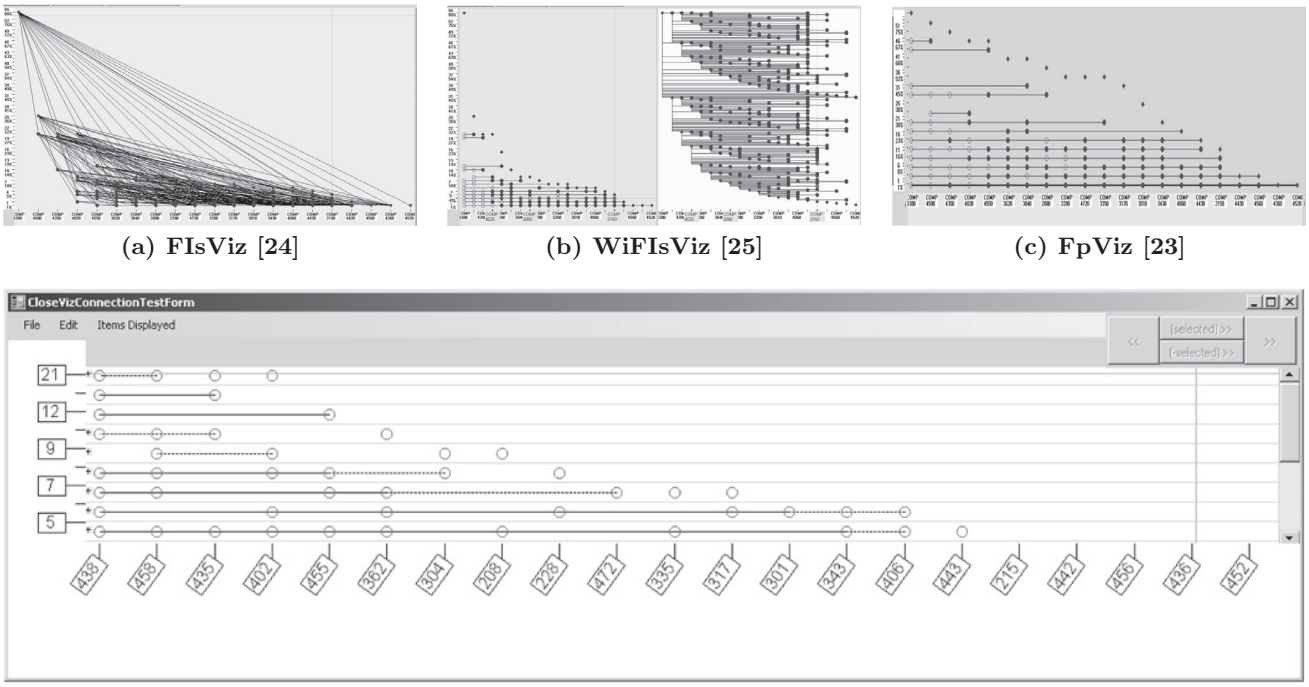
To test the expressiveness of our visualization, we formulated some questions that require participants to perform some level of analytical reasoning with the visualization. Examples of these questions include (i) Which course was most frequently taken (i.e., course with highest enrolment)? How many students were enrolled in that course? (ii) Which pairs of courses were frequently taken together? What were their frequencies? (iii) Which collections of k courses were frequently taken together by students, and what were their frequencies? (iv) To avoid exam hardship, which collections of three courses would one avoid scheduling within a time window of 24 hours?

We began the evaluation by presenting our CloseViz and asking the participants to explore it at their own will. We did not give them any information regarding what the symbols and representations meant in the visualization. We first questioned them on what they were able to identify. Evaluation results showed that, due to the simplicity of the representation of closed patterns by CloseViz, most of the participants were able to identify the basic meaning behind the representation (e.g., frequency was assigned to the y-axis, courses was assigned to the x-axis, circles denoted the items within useful patterns, [+] buttons only associated with dashed lines).

Afterwards, we gave the participants detailed information on how to read the graphs. Participants were then able to get a better understanding on what the solid and dashed lines meant. Evaluation results showed that a majority of the participants were able to correctly answer most of the questions. As expected, among the two groups of participants, the one with data mining background was more familiar with the concept of closed frequent pattern (which was new to participants without data mining background). Hence, participants with data mining background were all able to correctly derive frequent patterns using the shown closed patterns. This illustrated that *the closed patterns shown by CloseViz can be served as surrogates for the frequent patterns.*

5.3 Visual Exploration

In addition, we also asked the participants to try out other visualizers (FIsViz, WiFIsViz, FpViz). All participants liked CloseViz as it was much simpler (e.g., no need to distinguish the filled circles from unfilled ones) and it showed fewer lines (e.g., other three were quite crowded). Figure 7 shows snapshots of these four visualizers. As observed from Figure 7(a), FIsViz was crowded—especially on the bottom portion of the screen. It was not easy to visualize useful patterns (especially those with low frequencies). Moreover, polylines were bent and crossed over each other, it was not easy to trace the line sectors of some interesting patterns. As shown in Figure 7(b), the use of horizontal lines in WiFIsViz (instead of polylines) made it easier to trace the line sectors of some interesting patterns. However, the displayed information was packed as it used a half-screen to show the frequent patterns and another half-screen to show their frequencies. Participants needed to either (i) click on a line of a particular frequency (on the left half-screen, which triggered WiFIsViz to highlight the corresponding patterns on the right half-screen) to find out all patterns of that frequency or to (ii) click on a specific frequent pattern (on the right half-screen, which triggered WiFIsViz to highlight the corresponding horizontal line on the left half-screen) to find out the frequency of that pattern. In either case, it was not a straightforward task. Observed from Figure 7(c) that FpViz used the full-screen for showing the frequencies of patterns. However, every horizontal line was solid and was accompanied with [+] buttons. Participants did not know which line should be expanded and which should not. In contrast, as shown in Figure 7(d), some lines (e.g., for frequency 12) did not accompany with [+] buttons as these solid lines do not require expansion. Only the dashed lines, which accompanied with [+] buttons, required expansion. Moreover, fewer lines were shown.



(a) FIsViz [24] **(b) WiFIsViz [25]** **(c) FpViz [23]**

(d) Our proposed CloseViz

Figure 7: Snapshots of the four visualizers showing useful patterns mined from a student-course database.

Next, we evaluated the functionality of CloseViz. When compared CloseViz (which only shows closed patterns) with FIsViz, WiFIsViz and FpViz (which show all frequent patterns), all four visualizers provide users with the same information. As closed patterns are concise representation of all frequent patterns (i.e., with no information loss), users can obtain frequencies of all non-closed frequent patterns from the collection of closed patterns.

We also evaluated the performance of CloseViz: (i) We varied the size of the datasets. We measured the time both for mining closed patterns and for constructing the display layout (by using our proposed CloseViz). The results showed that the runtime (which includes CPU and I/Os) increased linearly with the number of transactions in the dataset. (ii) We varied the number of items in the domain. The results showed that the runtime increased when the number of domain items increased. (iii) We also varied the user-defined frequency threshold *minsup*. When *minsup* increased, the number of patterns that satisfy the threshold (i.e., frequent patterns to be displayed) decreased, which in turn led to a decrease in runtime.

6. CONCLUSIONS

Frequent pattern mining is an important aspect of KDD. Many mining algorithms return a collection of the mined patterns in the form of a textual list of frequent patterns. This list can be very long and difficult to comprehend. As "a picture is worth a thousand words", it is desirable to have visualization systems. However, many existing visualization systems were not designed to show frequent patterns. For those that were designed to do so, they display every frequent pattern. When handling very large datasets, the number of frequent patterns to be displayed can be huge due to pattern explosion. To improve this situation, we proposed a simple yet powerful *closed frequent pattern visualizer* called *CloseViz*, which provides users with explicit and easily-visible information among the closed patterns. Specifically, it represents closed patterns as horizontal lines in a two-dimensional graph. CloseViz greatly reduces the number of displayed patterns without losing any frequency information. As such, the displayed closed patterns mined from a dataset can be served as surrogates for all frequent patterns that can be mined from the same dataset. Moreover, CloseViz also provides users with interactive features for visual exploration. Evaluation results showed the usefulness of CloseViz in visualizing useful patterns in the form of closed patterns.

As ongoing work, we are conducting more extensive experiments to evaluate CloseViz. We are also investigating alternative orderings of domain items in the x-axis of CloseViz to see if an intelligent ordering would optimize the layout of horizontal lines. As future work, we would like to explore alternative representations (e.g., Cartesian contour [14], conditional profile summary [31]) of other useful patterns (e.g., approximate closed itemsets [7], fault-tolerant frequent patterns [32]).

7. ACKNOWLEDGMENTS

This project is partially supported by Natural Sciences and Engineering Research Council of Canada (NSERC) in the form of research grants.

8. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. VLDB 1994*, pp. 487–499.

[2] C. Ahlberg. Spotfire: an information exploration

environment. *ACM SIGMOD Record*, **25**(4), pp. 25–29, 1996.
[3] M. Ankerst, C. Elsen, M. Ester, and H.-P. Kriegel. Visual classification: an interactive approach to decision tree construction. In *Proc. KDD 1999*, pp. 392–396.
[4] S. Berchtold, H.V. Jagadish, and K.A. Ross. Independence diagrams: a technique for visual data mining. In *Proc. KDD 1998*, pp. 139–143.
[5] J. Blanchard, F. Guillet, and H. Briand. Interactive visual exploration of association rules with rule-focusing methodology. *KAIS*, **13**(1), pp. 43–75, 2007.
[6] C. Brunk, J. Kelly, and R. Kohavi. MineSet: an integrated system for data mining. In *Proc. KDD 1997*, pp. 135–138.
[7] H. Cheng, P.S. Yu, and J. Han. AC-Close: efficiently mining approximate closed itemsets by core pattern recovery. In *Proc. IEEE ICDM 2006*, pp. 839–844.
[8] C.H. Chih and D.S. Parker. The persuasive phase of visualization. In *Proc. KDD 2008*, pp. 884–892.
[9] B. Goethals, W. Le Page, M. Mampaey. Mining interesting sets and rules in relational databases. In *Proc. ACM SAC 2010*, pp. 997–1001.
[10] J. Han and N. Cercone. RuleViz: a model for visualizing knowledge discovery process. In *Proc. KDD 2000*, pp. 244–253.
[11] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: a frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, **8**(1), pp. 53–87, 2004.
[12] H. Hofmann, A.P.J.M. Siebes, and A.F.X. Wilhelm. Visualizing association rules with interactive mosaic plots. In *Proc. KDD 2000*, pp. 227-235.
[13] T. Iwata, T. Yamada, and N. Ueda. Probabilistic latent semantic visualization: topic model for visualizing documents. In *Proc. KDD 2008*, pp. 363–371.
[14] R. Jin, Y. Xiang, and L. Liu. Cartesian contour: a concise representation for a collection of frequent sets. In *Proc. KDD 2009*, pp. 415–425.
[15] D.A. Keim. Information visualization and visual data mining. *IEEE TVCG*, **8**(1), pp. 1–8, 2002.
[16] D.A. Keim and H.-P. Kriegel. Visualization techniques for mining large databases: a comparison. *IEEE TKDE*, **8**(6), pp. 923–938, 1996.
[17] D.A. Keim and J. Schneidewind (eds.). Special issue on visual analytics. *ACM SIGKDD Explorations*, **9**(2), 2007.
[18] Y. Koren and D. Harel. A two-way visualization method for clustered data. In *Proc. KDD 2003*, pp. 589–594.
[19] L.V.S. Lakshmanan, C.K.-S. Leung, and R.T. Ng. Efficient dynamic mining of constrained frequent sets. *ACM TODS*, **28**(4), pp. 337–389, 2003.
[20] H. Lam, D. Russell, D. Tang, and T. Munzner. Session viewer: visual exploratory analysis of web session logs. In *Proc. IEEE VAST 2007*, pp. 147–154.
[21] C. K.-S. Leung. Frequent itemset mining with constraints. *Encyclopedia of Database Systems*, pp. 1179–1183, 2009.
[22] C.K.-S. Leung and C.L. Carmichael. FpVAT: a visual analytic tool for supporting frequent pattern mining. *ACM SIGKDD Explorations*, **11**(2), pp. 39–48, 2009.
[23] C.K.-S. Leung and C.L. Carmichael. FpViz: a visualizer for frequent pattern mining. In *Proc. VAKD 2009*, pp. 30–39.
[24] C.K.-S. Leung, P.P. Irani, and C.L. Carmichael. FIsViz: a frequent itemset visualizer. In *Proc. PAKDD 2008*, pp. 644–652.
[25] C.K.-S. Leung, P.P. Irani, and C.L. Carmichael. WiFIsViz: effective visualization of frequent itemsets. In *Proc. IEEE ICDM 2008*, pp. 875–880.
[26] C.K.-S. Leung, Q.I. Khan, Z. Li, and T. Hoque. CanTree: a canonical-order tree for incremental frequent-pattern mining. *KAIS*, **11**(3), pp. 287–311, 2007.
[27] C.K.-S. Leung, R.T. Ng, and H. Mannila. OSSM: a segmentation approach to optimize frequency counting. In *Proc. IEEE ICDE 2002*, pp. 583–592.
[28] T. Munzner, Q. Kong, R.T. Ng, J. Lee, J. Klawe, D. Radulovic, and C.K.-S. Leung. Visual mining of power sets with large alphabets. Technical report UBC CS TR-2005-25, Department of Computer Science, The University of British Columbia, Vancouver, BC, Canada, 2005.
[29] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. ICDT 1999*, pp. 398–416.
[30] J. Pei, J. Han, and R. Mao. Closet: an efficient algorithm for mining frequent closed itemsets. In *Proc. ACM SIGMOD Workshop on DMKD 2000*, pp. 21–30.
[31] A.K. Poernomo and V. Gopalkrishnan. CP-summary: a concise representation for browsing frequent itemsets. In *Proc. KDD 2009*, pp. 687–696.
[32] A.K. Poernomo and V. Gopalkrishnan. Towards efficient mining of proportional fault-tolerant frequent itemsets. In *Proc. KDD 2009*, pp. 697–706.
[33] G. Pölzlbauer, A. Rauber, and M. Dittenbach. A vector field visualization technique for self-organizing maps. In *Proc. PAKDD 2005*, pp. 399–409.
[34] K. Puolamäki and A. Bertone (eds.). Special issue on visual analytics and knowledge discovery. *ACM SIGKDD Explorations*, **11**(2), 2009.
[35] C. Stolte, D. Tang, and P. Hanrahan. Query, analysis, and visualization of hierarchically structured data using Polaris. In *Proc. KDD 2002*, pp. 112–122.
[36] N. Tatti. Maximum entropy based significance of itemsets. In *Proc. IEEE ICDM 2007*, pp. 312–321.
[37] N. Tatti and J. Vreeken. Finding good itemsets by packing data. In *Proc. IEEE ICDM 2008*, pp. 588–597.
[38] P.C. Wong and J. Thomas. Visual analytics. *IEEE CG&A*, **24**(5), pp. 20–21, 2004.
[39] L. Yang. Pruning and visualizing generalized association rules in parallel coordinates. *IEEE TKDE*, **17**(1), pp. 60–70, 2005.
[40] X. Yang, S. Asur, S. Parthasarathy, and S. Mehta. A visual-analytic toolkit for dynamic interaction graphs. In *Proc. KDD 2008*, pp. 1016–1024.
[41] J. Yuan, Y. Wu, and M. Yang. From frequent itemsets to semantically meaningful visual patterns. In *Proc. KDD 2007*, pp. 864–873.
[42] M.J. Zaki. Closed itemset mining and non-redundant association rule mining. *Encyclopedia of Database Systems*, pp. 365–368, 2009.
[43] M.J. Zaki and C.-J. Hsiao. CHARM: an efficient algorithm for closed itemset mining. In *Proc. SDM 2002*, pp. 457–473.

A framework for mining interesting pattern sets

Tijl De Bie
University of Bristol, Intelligent Systems Laboratory
Merchant Venturers Building, Bristol, BS8 1UB, UK
tijl.debie@gmail.com

Kleanthis-Nikolaos Kontonasios
University of Bristol, Intelligent Systems Laboratory
Merchant Venturers Building, Bristol, BS8 1UB, UK
kk8232@bristol.ac.uk

Eirini Spyropoulou
University of Bristol, Intelligent Systems Laboratory
Merchant Venturers Building, Bristol, BS8 1UB, UK
enxes@bristol.ac.uk

ABSTRACT

This paper suggests a framework for mining subjectively interesting pattern sets that is based on two components: (1) the encoding of prior information in a model for the data miner's state of mind; (2) the search for a pattern set that is maximally informative while efficient to convey to the data miner.

We illustrate the framework with an instantiation for tile patterns in binary databases where prior information on the row and column marginals is available. This approach implements step (1) above by constructing the MaxEnt model with respect to the prior information [2, 3], and step (2) by relying on concepts from information and coding theory.

We provide a brief overview of a number of possible extensions and future research challenges, including a key challenge related to the design of empirical evaluations for subjective interestingness measures.

Categories and Subject Descriptors

H.2.8 [**Database management**]: Database applications—*Data mining*; I.5.1 [**Pattern recognition**]: Models—*Statistical*

Keywords

Subjective interestingness measures, pattern set mining, prior information, maximum entropy.

1. BACKGROUND

Motivation.

Since the introduction of the Apriori algorithm significant progress has been made in developing increasingly efficient and sophisticated frequent itemset mining algorithms. Today, we have arguably reached the point where progress in this respect has become incremental. Perhaps to a lesser extent the same holds for other pattern mining techniques.

UP'10, July 25th, 2010, Washington, DC, USA.

Instead, in real-life applications of pattern mining new challenges have surfaced (e.g. [15]). Most of these are centered around the observed discrepancy between what is intuitively interesting and the existing objective proxies, such as the frequency of a pattern. Indeed, a strong consensus is growing that finding better objective formalizations of what is intuitively or subjectively interesting is critical for the success of the field. A second problem is that the set of all patterns deemed interesting by any specific interestingness measure contains too many patterns to be convenient for human consumption, many of which are highly redundant.

Matching these two problems, two research avenues are being pursued by the research community. The first problem is addressed by the search for better interestingness measures, even if that comes at an added computational cost when compared to monotonic or anti-monotonic measures (see [8] for an overview). The second problem is addressed by searching for interesting *pattern sets*, rather than for sets of interesting patterns (see e.g. [4, 1, 7, 5, 6]). These lines of research are by no means independent, and many methods attempt to address them simultaneously.

Despite significant recent progress on both these fronts, we believe the challenges cannot be fully met by proposing yet more objective measures with new properties. Indeed, counting only probabilistically inspired objective interestingness measures, in the survey paper [8] the authors list 38 of them. It is clear that expanding this zoo of interestingness measures even further would not increase transparency of the research area. Instead, the recent advances need to be embedded in a flexible and interactive approach.

In this paper, we discuss a framework that tries to achieve this, which we believe addresses both problems discussed above. It relies on formalizing the prior information of the data miner, and contrasting the data with this formal representation of the state of mind of the data miner. In this way, a pattern set can be found that is subjectively interesting, and data mining algorithms become intelligent communication interfaces between the data and the data miner.

Below we mostly consider patterns (such as itemset and tile patterns) in binary databases. However, we wish to stress that our framework is more widely applicable, and we will therefore introduce it in general terms.

Subjective interestingness measures.

The distinction between subjective and objective interestingness measures was first made in [21, 18, 19], and adopted in the survey paper [8]. Subjective interestingness measures as they conceive them are interestingness measures that do

not only depend on properties of the pattern, but also on the class of users of the data mining algorithm. They should quantify at least one of two properties: unexpectedness, or actionability. Both are clearly strongly dependent on the data miner's prior information or goals.

The first attempt at designing a subjective interestingness measure quantifying unexpectedness was made by [21]. They made use of a so-called belief system, which consists of a set of rules with associated degrees of belief, representing what the data miner knows about the data. Then, patterns are deemed more interesting if they strongly affect these beliefs in a Bayesian sense. The approach had some drawbacks, the most important of which is probably that interactions between these rules were hard to control: patterns implied by combinations of rules from the belief system would be deemed unexpected by their system, if they are not implied by any single rule by itself. While the practical implication of this work is perhaps limited for these reasons, its importance as a conceptual breakthrough is hard to overestimate.

Still, since this work, very few other subjective interestingness measures have been proposed (see [8] for an overview). The most promising one is probably from [12], where the authors suggest to model transactions in a binary database using Graphical Models, and use this model to compute the expected support of itemsets. Itemsets are then deemed more interesting if their support deviates more strongly (in absolute sense) from the expected support given this model. The Graphical Model could be designed such that it reflects the prior information of a data miner, although it is less clear how to do this in practice (where data miners may have limited expertise in Graphical Models). Furthermore, it assumes that transactions are independent and identically distributed, often false in practice. And lastly, the absolute difference between expected and observed support may not be the best measure of unexpectedness.

Some other recent approaches claim to take into account prior information to quantify interestingness, such as [5, 6, 9, 17, 11]. All of these are based on hypothesis testing to formalize interestingness, where the null hypothesis is designed to represent the prior information of the data miner. Those based on randomization approaches [9, 17, 11] are computationally demanding but are also more flexible. Still, they are limited to specific types of prior information, such as on row and column marginals [9, 17], and more recently also cluster structure and the frequency of given itemsets [11].

The framework proposed in this paper aims to be flexible, as well as realistically useful in real-life data mining settings.

2. A GENERAL FRAMEWORK

Below we will first introduce the rationale of our framework. Then we will detail the two basic components: modeling the prior information, and searching for pattern sets that are interesting when contrasted to this prior information.

2.1 Shifting the focus: from the data to the miner

In designing objective interestingness measures, a data mining researcher tries to enter the mind of an imagined practitioner, and attempts to rationalize what may be intuitively of interest to this practitioner. This approach has born fruit in two respects. First, it has helped in understanding which types of interestingness measures are amenable to efficient algorithms. Second, for specific applications, special-purpose interestingness measures are often desirable.

However, the strategy of entering a specific practitioner's mind inevitably falls short of the design of measures that can be applied in a wide range of circumstances, by a wide range of practitioners. In the design of flexible subjective measures, we believe it essential to consider the data mining practitioner as the object of study, no less than the data itself. Such a Copernican revolution, shifting the focus from the data to the data mining process (Fig. 1), is likely to be necessary if we intend to capture subjectivity. Indeed, this can only be achieved if the algorithm is aware of what the data miner wants or does not want to learn about the data.

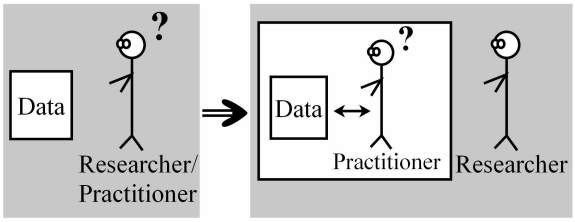


Figure 1: The researcher's point of view when designing objective interestingness measures (left, where he coincides with the practitioner) and subjective interestingness measures (right).

In this paper, we aim to formalize the data mining process as we envisage it with the above considerations in mind. We do this by explicitly modeling the data miner's prior information and hence what is *not* interesting to him. What is of interest to the data miner is then what contrasts with this prior information. Hence, in the actual mining step, a set of patterns is sought that is maximally interesting given the prior information. We believe this approach safeguards the exploratory nature of pattern mining methods better than a more limiting approach that directly specifies what is interesting.

Thus, there are two essential components in our framework: the modeling of the data miner's prior information, and the subsequent search for a pattern set of which the occurrence in the data contrasts with this model of prior information. Below we will fill in this framework more concretely, detailing both these aspects individually.

Throughout this Section, we will complement the theory with an example for the case of binary databases represented by the binary matrix $\mathbf{D} \in \{0,1\}^{m \times n}$, where the prior information is on row and column marginals $\sum_j \mathbf{D}(i,j)$ and $\sum_i \mathbf{D}(i,j)$, and where we are searching for interesting tiles defined by a subset of rows $I \subseteq \{1, \cdots, m\}$ and a subset of columns $J \subseteq \{1, \cdots, n\}$ such that $\mathbf{D}(i,j) = 1$ for all $i \in I$ and $j \in J$. This example is given for concreteness only, and in Sec. 3.3 we aim to make it clear that it can be applied much more generally.

In this paper, random variables will be underlined (e.g. $\underline{\mathbf{D}}$), while deterministic samples of these random variables are not underlined (e.g. $\mathbf{D}$ is a specific instance of the data).

2.2 Formalizing prior information in a probabilistic model

As suggested in [2, 3], we choose to formalize the prior in-

formation in a probability distribution P defined over the data space $\mathcal{D}$. This can be done by setting up a probabilistic model for the data $\mathbf{D}$ that satisfies certain constraints imposed by the prior information. Note that typically, the data $\mathbf{D}$ itself is composed of a set of variables: $\mathbf{D} = \{\mathbf{d}_k, k = 1 : n\}$ with typically $\mathbf{d}_k \in \{0, 1\}$ or $\mathbf{d}_k \in \mathbb{N}$, or $\mathbf{d}_k \in \mathbb{R}$ (e.g. the entries in a database).

The type of prior information we will consider is in the form of expectations about certain functions f_i (further called constraints functions) of the data $\mathbf{D}$:

$$E_P\{f_i(\underline{\mathbf{D}})\} \quad = \quad c_i.$$

EXAMPLE 2.1. *As an example for a binary database $\mathbf{D}$, we will consider two classes of constraint functions, computing the row and the column marginals of the database: $f_i^r(\mathbf{D}) \triangleq \sum_j \mathbf{D}(i,j)$ and $f_j^c \triangleq \sum_i \mathbf{D}(i,j)$. I.e., the constraints are:*

$$\begin{aligned} E_P\left\{\sum_j \underline{\mathbf{D}}(i,j)\right\} &= c_i^r, \\ E_P\left\{\sum_i \underline{\mathbf{D}}(i,j)\right\} &= c_j^c, \end{aligned}$$

where c_i^r and c_j^c are the required expected row and column marginals. This means that we assume that the data miner has certain expectations on each of the row and column marginals as prior information.

Using constraints on the expectations of certain properties of the data, as quantified by the functions f_i, is a flexible way of encoding prior information. We will give more examples in Sec. 3.3.[1]

As in practical settings prior information will not be so rich as to uniquely determine the distribution, an inductive bias needs to be chosen. For various reason discussed in [2, 3] and references therein, it makes sense to choose the distribution of maximum entropy among all those that satisfy the constraints:

$$\begin{aligned} \max_P \quad & -E_P\{\log(P(\underline{\mathbf{D}}))\}, \\ \text{s.t.} \quad & E_P\{f_i(\underline{\mathbf{D}})\} = c_i, \\ & P(\mathbf{D}) \geq 0 \ \forall \mathbf{D} \in \mathcal{D}, \\ & \sum_{\mathbf{D}\in\mathcal{D}} P(\mathbf{D}) = 1. \end{aligned}$$

We refer to the resulting problem as the MaxEnt model.

It is well-known (and easy to prove using Lagrange duality theory) that the solution of the maximum entropy optimization problem takes the form of an exponential family distribution (see e.g. [23]):

$$P(\mathbf{D}) = \frac{1}{Z(\boldsymbol{\lambda})} \exp\left(\sum_i \lambda_i f_i(\mathbf{D})\right),$$

where $\boldsymbol{\lambda}$ denotes a vector containing all λ_i and $Z(\boldsymbol{\lambda}) = \sum_{\mathbf{D}\in\mathcal{D}} \exp\left(\sum_i \lambda_i f_i(\mathbf{D})\right)$ is known as the partition function and ensures normalization. The values of the Lagrange multipliers λ_i can be found by solving the dual optimization problem, which is formally identical to minimizing the negative log-likelihood of data $\mathbf{D}$ that satisfies the constraints $f_i(\mathbf{D}) = c_i$ exactly. Mathematically, this optimization problem is written as:

$$\min_{\boldsymbol{\lambda}} \quad \log(Z(\boldsymbol{\lambda})) - \sum_i \lambda_i c_i.$$

It is worth emphasizing that the exponential family of distributions encompasses most widely used distributions, including the Bernoulli, binomial, Poisson, and Gaussian distributions and many others.

EXAMPLE 2.2. *The MaxEnt model for prior information on row and column marginals on a binary database $\mathbf{D}$ as defined in Ex. 2.1 is given by an exponential family distribution that can be rewritten as a product distribution of Bernoulli random variables, one for each database entry:*

$$\begin{aligned} P(\mathbf{D}) &= \prod P_{ij}(\mathbf{D}(i,j)), \\ P_{ij}(\mathbf{D}(i,j)) &= \begin{cases} \frac{\exp(\lambda_i^r+\lambda_j^c)}{1+\exp(\lambda_i^r+\lambda_j^c)} & \text{if} \quad \mathbf{D}(i,j) = 1, \\ \frac{1}{1+\exp(\lambda_i^r+\lambda_j^c)} & \text{if} \quad \mathbf{D}(i,j) = 0. \end{cases} \end{aligned}$$

Note that although the random variables for the different database entries are independent, their distributions are related by the parameters λ_i^r for the rows and λ_j^c for the columns. These are obtained by solving the optimization problem:

$$\min_{\boldsymbol{\lambda}^r,\boldsymbol{\lambda}^c} \quad \sum_{i,j} \log(1 + \exp(\lambda_i^r + \lambda_j^c)) - \sum_i \lambda_i^r c_i^r - \sum_j \lambda_j^c c_j^c.$$

It is shown in [3] that this problem can be solved remarkably efficiently even for very large databases.

The duality relation between the Maximum Entropy and Maximum Likelihood problems, with the exponential family as a hinge between them, is well known in mathematical statistics. Also in the Graphical Models literature, it has been studied for the special case where the constraint functions f_i are so-called potential functions, i.e. (often indicator) functions that pertain to a typically small subset of the variables $\mathbf{d}_k$ making up the data $\mathbf{D}$. (See [23] for an overview.) In this context, however, we do not constrain ourselves to this situation. Allowing more general functions leads to models such as in Ex. 2.2 where the graphical model representation would be trivial (all random variables $\underline{\mathbf{d}}_i$ making up the data $\underline{\mathbf{D}}$ are independent), but where the distributions of these random variables are related by sharing certain parameters in a non-trivial way.

2.3 Information theory to quantify subjective interestingness

Given a probabilistic model capturing the prior information about the data, we can now attempt to quantify the interestingness to the data miner of a given pattern in the data. Taking account of the prior information, such quantification will be inherently subjective.

Formalizing patterns.

Before we can proceed, we need to define formally what we mean by a pattern.

DEFINITION 2.3. *Let $\pi : \mathcal{D} \to \mathbb{R}$ be a function that we call a* pattern function *and that is an element from the* pattern

[1] Note that hard constraints of the form $g(\mathbf{D}) = c$ can also be imposed in this way, by using an indicator function $f_i(\mathbf{D}) \triangleq \delta(g(\mathbf{D}) = c)$. Then, $g(\mathbf{D}) = c$ with probability one if $E_P\{f_i(\underline{\mathbf{D}})\} = 1$ is imposed as a constraint.

space Π, *i.e.* $\pi \in \Pi$. *A* pattern *in the data* $\mathbf{D}$ *is defined as an equality of the form:*

$$\pi(\mathbf{D}) = \hat{\pi}.$$

We call $\hat{\pi} \in \mathbb{R}$ *the* pattern strength.

For example, in the context of frequent itemset mining, the pattern functions π considered are functions that evaluate as the frequency of an itemset. The set Π of all such functions is determined by the collection of all frequent itemsets. This definition is different from the standard definition in frequent pattern mining: the pattern for us is not the recurring element, but the fact that the element recurs a certain number of times in the data (as expressed by the equality $\pi(\mathbf{D}) = \hat{\pi}$). Let us give another example:

EXAMPLE 2.4. *We define the pattern functions as indicator functions for the presence of a tile, and denote them as* $\pi_{I,J}$ *for a tile with rows* I *and columns* J*. I.e.:*

$$\pi_{I,J}(\mathbf{D}) = \begin{cases} 1 & \text{if } \forall i \in I, j \in J : \mathbf{D}(i,j) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Hence, the pattern strength for a pattern function $\pi_{I,J}$ *is equal to* 1 *if the tile* (I, J) *is present in the data, and* 0 *otherwise.*

We believe the risk associated with using a non-standard definition for a pattern is outweighed by an important benefit: it allows us to deal with a much broader class of problems than just frequent pattern mining. As a result, the framework described in this paper can be transferred easily to other types of patterns, such as tile patterns (see Ex. 2.4), clustering patterns, classification patterns, etc.

The self-information of a pattern.

Before defining the interestingness of a pattern $\pi(\mathbf{D}) = \hat{\pi}$, we need to quantify the amount of information in the pattern as perceived by the data miner. This is adequately formalized by the Shannon self-information of the pattern with respect to distribution P that formalized the prior information, defined as the negative log-probability of seeing the observed pattern strength. Formally:

DEFINITION 2.5. *The* self-information *of a pattern* $\pi(\mathbf{D}) = \hat{\pi}$ *is defined as:*

$$I(\pi, \hat{\pi}) = -\log\left(Pr(\pi(\underline{\mathbf{D}}) = \hat{\pi})\right).$$

It is equal to the number of bits required to encode the pattern strength $\hat{\pi}$ *of this pattern under a Shannon optimal code with respect to the MaxEnt distribution* P *for* $\underline{\mathbf{D}}$*.*

The self-information is known to be the code length of a random variable (here the pattern strength $\hat{\pi}$ in the data $\mathbf{D}$) under a Shannon-optimal code with respect to the distribution P. Hence, this quantity also has an interpretation in terms of description length.

EXAMPLE 2.6. *Under the MaxEnt model from Ex. 2.2 and with the tile pattern functions* $\pi_{I,J}$ *from Ex. 2.4, the self-information of a pattern* $\pi_{I,J}(\mathbf{D}) = 1$ *is defined as:*

$$\begin{aligned} I(\pi_{I,J}, 1) &= -\log\left(\prod_{i \in I, j \in J} P_{ij}(1)\right), \\ &= -\sum_{i \in I, j \in J} \log\left(\frac{\exp(\lambda_i^r + \lambda_j^c)}{1 + \exp(\lambda_i^r + \lambda_j^c)}\right). \end{aligned}$$

With the MaxEnt model as a representation of the practitioner's uncertainty about the data, the self-information is equal to the information conveyed to the data miner when he is informed about the fact that a certain tile is present rather than not present in the data $\mathbf{D}$*. It is equal to the required code length if the presence of the tile is described using a Shannon optimal code with respect to the MaxEnt distribution. The self-information would be larger if, given the prior information, the tile is less likely to be present.*

The description length of a pattern.

The self-information $I(\pi, \hat{\pi})$ is the amount of information transmitted to the data miner if he is made aware of the presence of the pattern $\pi(\mathbf{D}) = \hat{\pi}$. The question now arises what the true cost is of communicating this information to the data miner. Can this be done more efficiently than with a Shannon-optimal code with respect to the MaxEnt model? It is clear that this is only possible if there are patterns in the data that are not to be expected given the MaxEnt model of the prior information, and we argue these are precisely the ones the data miner is interested in.

The true cost of conveying a pattern can be quantified by establishing a coding scheme to encode pattern functions $\pi \in \Pi$, and similarly for the pattern strengths $\hat{\pi}$. Then the cost can be defined as the description length $D(\pi, \hat{\pi})$ of the pattern $\pi(\mathbf{D}) = \hat{\pi}$ in this coding scheme. The coding scheme should be chosen so that it reflects the perceived complexity of a pattern. This approach based on coding lengths is convenient, as it will allow us to compare like with like when contrasting this cost with the self-information.

A difficulty with this approach is the design of a code, which can be cumbersome. As a shortcut, however, one could just specify the code lengths directly. When doing so, they must be such that, in principle, a uniquely decipherable code exists with these code word lengths. This means that the code words must satisfy Kraft's inequality [14].

A set of code lengths satisfying Kraft's inequality can be designed conveniently by first defining a distribution Q over the set of patterns that may need to be encoded, and choosing the code lengths of all patterns equal to their negative log-probability under that distribution. (Note that the obtained code lengths may not be integers, but they can still be achieved in the limit if a large number of these patterns are to be encoded using a Shannon-optimal coding under that chosen distribution.)

It should be stressed that the distribution Q and the description length are unrelated to the prior information the data miner holds about the data, and they are also unrelated to any stochastic process from which the data may have been sampled. It is no more than a mathematical construct to help the data miner in quantifying how hard it is for him to grasp a given pattern.

DEFINITION 2.7. *The* description length $D(\pi, \hat{\pi})$ *of a pattern* $\pi(\mathbf{D}) = \hat{\pi}$ *is given by its description length in a code chosen by the data miner, capturing the complexity of patterns as he perceives it.*

It is convenient to compute the description length indirectly, by first specifying a distribution Q *over the space of possible pairs* $(\pi, \hat{\pi})$*. Then the description length of a pattern* $\pi(\mathbf{D}) = \hat{\pi}$*, is given by the negative log-probability of* $(\pi, \hat{\pi})$ *under distribution* Q*:*

$$D(\pi, \hat{\pi}) = -\log(Q(\pi, \hat{\pi})).$$

EXAMPLE 2.8. *To encode a tile (I, J), for each row i and for each column j we need to specify whether or not $i \in I$ and $j \in J$. We will design a coding scheme for tiles using the approach above, i.e. by first defining a distribution Q.*

Let us assume that a data miner finds a tile easier to grasp if it contains less rows and less columns, with no distinction made between different rows or columns. Then, the distribution Q could be defined as:

$$\begin{aligned} Q(\pi_{I,J}, 1) &= p^{|I|+|J|}(1-p)^{m+n-|I|-|J|}q, \\ Q(\pi_{I,J}, 0) &= p^{|I|+|J|}(1-p)^{m+n-|I|-|J|}(1-q), \end{aligned}$$

where the $0 \leq p, q \leq 1$, p is the probability that any row or column belongs to a tile, and q is the probability of a pattern strength equal to 1. After some calculations, this means that the description length of a tile pattern with $\hat{\pi} = 1$ is equal to:

$$D(\pi_{I,J}, 1) = C + (|I| + |J|)D,$$

where $C = -(m+n)\log(1-p) - \log(q)$ and $D = \log\left(\frac{1-p}{p}\right)$.

For $p > 0.5$, it holds that $D > 0$, and the description length increases linearly with the circumference of the tile. The parameter p allows the data miner to zoom in to small tiles (smaller p), or zoom out to larger tiles (larger p). In the experiments below, we chose p equal to the density of the database, i.e. equal to the probability that a randomly selected row contains a 1 in a randomly selected column. We further chose q equal to 1, such that only pattern strengths equal to 1 would be considered.

The interestingness of a pattern.

The interestingness of a pattern can now be determined by comparing the description length of the pattern with its information content. In particular, we suggest to define the interestingness of a pattern as follows:

DEFINITION 2.9. *The* interestingness *of a pattern $\pi(\mathbf{D}) = \hat{\pi}$ is defined as the ratio of the self-information over the description length:*

$$interestingness(\pi, \hat{\pi}) = \frac{I(\pi, \hat{\pi})}{D(\pi, \hat{\pi})}.$$

Intuitively speaking, this quantifies the compression ratio of the information in the pattern by reporting it as a pattern in the code representing the data miner's intuition of simplicity.

EXAMPLE 2.10. *For the tile example, the interestingness measure will be larger if it covers as many (improbable) entries as possible (i.e. if it has a large surface), while having a circumference that is as small as possible.*

Interesting pattern sets.

In practice, a data miner will rarely be satisfied with just the single most interesting pattern. It is likely that the data miner has a certain finite processing capability, determining an upper bound u on the total description length of all patterns reported. Given this upper bound, the data miner would like to receive as much information as possible when contrasted with his prior information. This information can be captured adequately by the self-information of the pattern set, defined as:

DEFINITION 2.11. *The* self-information *of a pattern set is defined as the negative log-probability that these patterns are present in the data under the MaxEnt model. Formally, with pattern functions $\pi \in \Pi_s \subseteq \Pi$ and associated pattern strengths $\hat{\pi} = \pi(\mathbf{D})$ observed in the data $\mathbf{D}$:*

$$I\left(\{(\pi, \hat{\pi}), \pi \in \Pi_s\}\right) = -\log\left(Pr(\pi(\underline{\mathbf{D}}) = \hat{\pi}, \forall \pi \in \Pi_s)\right)$$

with respect to the MaxEnt distribution for $\underline{\mathbf{D}}$.

To maximally satisfy the data miner, the data mining algorithm should thus solve the following optimization problem:

$$\begin{aligned} \max_{\Pi_s \subseteq \Pi} \quad & I\left(\{(\pi, \hat{\pi}), \pi \in \Pi_s\}\right), \\ \text{s.t.} \quad & \sum_{\pi \in \Pi_s} D(\pi, \hat{\pi}) \leq u. \end{aligned}$$

The most interesting pattern set subject to the imposed constraint on the description length is then defined by the optimal set of pattern functions Π_s.

This optimization problem is unfortunately a combinatorial one, and it is hard to solve in general. However, in some cases it may be easy to solve it or at least to solve it approximately. Let us illustrate this with an example.

EXAMPLE 2.12. *The self-information of a pattern set with tile-patterns $\pi_{I,J} = 1$ with $\pi_{I,J} \in \Pi_s$ is given by:*

$$\begin{aligned} & I\left(\{(\pi_{I,J}, 1), \pi_{I,J} \in \Pi_s\}\right) \\ = & -\log\left(\prod_{i,j:\exists \pi_{I,J} \in \Pi_s : i \in I \& j \in J} P_{ij}(1)\right), \\ = & -\sum_{i,j:\exists \pi_{I,J} \in \Pi_s : i \in I \& j \in J} \log\left(\frac{\exp(\lambda_i^r + \lambda_j^c)}{1 + \exp(\lambda_i^r + \lambda_j^c)}\right). \end{aligned}$$

Hence, the problem can be phrased as follows. Given is the set of entries in the database and a collection of subsets of this set as covered by the tiles. Each entry has a certain weight ($-\log(P_{ij}(\mathbf{D}(i,j))$), and each subset has a certain cost ($D(\pi, \hat{\pi})$). The pattern set mining task can then be formulated as the search for a collection of subsets maximizing the sum of the weights of the entries in its union, subject to an upper bound on the sum of the costs of the subsets in the collection.

When the tiles in the database are precomputed using an existing itemset miner (e.g. CHARM), this is an instance of the weighted budgeted maximum coverage problem, which is NP-hard but can be solved approximately to an approximation ratio of $1 - \frac{1}{e}$ using a greedy algorithm. In this algorithm, the k'th tile pattern is selected as the one that maximizes the ratio of the sum of the weights of the newly covered entries divided by its description length.

We have applied this method to two abstract databases after stop-word removal and stemming (turned into binary databases by considering rows as texts and columns as words). The first dataset contains all KDD abstracts between 2001 and 2008, which amounts to 843 documents and 6154 unique stemmed words. The second dataset contains all ICDM abstracts up to 2007, amounting to 859 documents and 5006 unique stemmed words. The 15 tiles first selected in this greedy algorithm are shown in the left column of Tab. 1. Only tiles corresponding to closed itemsets and a support of at least 5 were considered (as mined by CHARM [24]).

KDD

Mining interesting pattern sets (current paper)	$\|I\|$	Tiling databases as described in [7]	$\|I\|$
machin support svm vector	25	data paper	389
art state	39	algorithm propos	246
labeled learn supervised unlabeled	10	data mine	312
associ mine rule	36	base method	202
express gene	25	result show	196
frequent itemset	28	problem	373
graph larg network social	15	data set	279
column row	13	approach	330
algorithm faster magnitud order	12	model	301
algorithm data paper propos real synthetic	27	present	296
answer question	18	larg	286
nearest neighbor	13	applic	271
classifi featur machin support text vector	9	perform	266
precis recal	14	real	255
decis tree	33	inform	240

ICDM

Mining interesting pattern sets (current paper)	$\|I\|$	Tiling databases as described in [7]	$\|I\|$
classifi machin support vector	24	algorithm data	338
analysi discriminant lda linear	10	paper propos	237
associ database mine rule	28	data mine	279
bayes naiv	23	show	370
algorithm discov frequent mine pattern	28	base	369
nearest neighbor	20	result	359
art state	22	approach	349
cluster data dimensional high subspace	11	method	346
account take	19	set	343
play role	14	problem	330
document text word	14	present	305
exampl learn train	17	perform	265
algorithm em expect maximization	8	model	239
frequent item itemset mine	18	larg	221
classifi decis tree	20	algorithm propos	271

Table 1: This table reports the sets of words (columns) J as well as the number of documents $|I|$ containing all these words for the top-15 selected tiles (I, J) for two methods: the tile-mining approach described in this paper (left column) and the tiling databases approach (right column).

3. DISCUSSION

Below we will first try to further elucidate our framework by providing interpretations and clarifying some of the choices we have made. Then we will first discuss some relations with prior work. Finally, we will provide an overview of the extensions and various instantiations of our framework that are subject of current work and that pose interesting challenges for future work.

3.1 Interpretations and remarks

The nature of the data.

In introducing the general framework, we have intentionally treated the data $\mathbf{D}$ as a monolitic block. Our intention with this is to emphasize that our focus is broader than *data sets*. Our framework should be able to handle data that cannot elegantly be cast in a set, such as networks or relational databases. A set often suggests that the elements are commensurable or comparable, perhaps even sampled i.i.d. from some distribution. In this paper, we do not want to make such suggestion or unrealistic assumptions.

The nature of the prior information.

The term prior information may be somewhat misleading, and perhaps more accurately we could also have chosen *prior expectations*. Indeed, the prior information may be wrong (if the data miner is ill-informed), and we believe our framework deals with this in an appropriate way. If the prior information is incorrect, patterns that correct for this will be flagged up as interesting, which is desirable in practice.

Another remark with regard to prior information is that it may seem impractical to list what the data miner already knows. However, we believe that in many cases the most important prior information can be stated at a meta-level, in general terms. For example, the prior information in our running example was on all row and column marginals, specifying $n + m$ constraints in a description of just a few words.

The code describing patterns.

There is a strong connection between a code to describe patterns, and a syntactic choice for the patterns. Indeed, fixing a syntax for the patterns is similar to fixing a code. Simpler patterns in the syntax are be easier to parse and hence probably easier to understand for a data miner.

A communication metaphor.

Our framework can be described using a communication metaphor between the data (Alice) and the data miner (Bob), whereby the data mining algorithm is the intermediary interfacing with both. See Fig. 2 for a graphical illustration. The goal in this communication protocol is to communicate the data as efficiently as possible (i.e. with the shortest possible description), by relying on any prior information the data miner may have. In the first instance good compression can be achieved by relying on a Shannon-optimal code with respect to the MaxEnt model specified by this prior information. However, if the data miner believes or hopes that patterns of a certain easily understandable syntactic form are present in the data, he may ask the Alice to communicate these separately, potentially reducing the overall coding length. In a data mining context, of course only the patterns would be sent, not the rest of the data.

3.2 Relations to prior work

Tiling databases.

The work on tiling databases [7] fits in most closely with our framework, and can be described as a specific instantiation of it. One of the goal in that work was to come up with a collection of a fixed number of tiles (the pattern set) that covers as many database entries as possible. They already observed that set covering techniques can be used to efficiently solve this problem to a guaranteed approximation ratio. The results on two textual datasets described above are shown in Tab. 1.

To see how this method can be viewed as an instance of our framework we need to specify two things: the prior information used, and the code for encoding the patterns. Let us first consider the prior information. Since each database entry is given the same weight, the prior information used is empty, or perhaps non-informative such as assuming that all row marginals are equal, and also all column marginals. As for the coding scheme for the patterns, the same cost is attributed to each of the tiles, such that no distinction is made between tiles with a small or a large circumference.

In this light, it is easy to understand the difference in output between the results of the tile mining method discussed in the running example and the tile mining method presented in [7]. Many tiles found by our method achieve a balance between number of words and documents, since encoding long stretched out tiles comes at a greater cost than compact square tiles. Furthermore, they are less susceptible to common uninformative words, preferring tiles with uncommon words (and although this is harder to see, also preferring tiles overlapping with shorter documents).

KRIMP.

Another related method is KRIMP [20], which attempts to describe the database by constructing a code table of itemsets and encoding the database by making use of this code table. Our approach bears some clear similarities to KRIMP, notably the reliance on coding and description length ideas. However, like with other objective interestingness measures, KRIMP seems less flexible in its current form, and there seems to be no direct way of incorporating properties of the data miner.

Maximum entropy based significance of itemsets.

The maximum entropy principle has been used before for the purpose of designing an interestingness measure for itemsets [22]. Here, the frequency of an itemset is contrasted with the expected frequency based on the frequencies of its subsets. To compute this expected frequency, maximum entropy modeling is used. While this is potentially useful in various applications, it is still an objective measure, that cannot be fine-tuned to suit particular data mining practitioners or tasks.

3.3 Extensions and further work

We are currently working on extending the above ideas in various ways, applying the framework to more general data types, for more general pattern types, and for more general types of prior information. Of course, there are significant interactions between these extensions, but for convenience let us discuss them one by one. After that, we will discuss some other interesting challenges for future work.

Ongoing work.

We have introduced our framework for general data $\mathbf{D}$, as we believe it is likely to be useful for data types different from just binary databases. A first possible extension is toward non-binary databases, such as categorical, integer-valued, and real-valued data (see also [2, 3]). More importantly, we are currently working on instantiating this framework in a flexible way for relational databases. This will allow us to mine for interesting patterns in relational databases in the spirit of the recent papers [16, 10], which have given a new and promising twist to pattern mining research.

Concerning the types of pattern, in a recent paper [13] we have discussed an instantiation of the framework for noisy tiles, with promising empirical results. Other extensions toward frequent itemsets might be of interest as well.

The prior information we have considered in the running example in this paper was restricted to the row and column marginals. Other types of prior informations we are currently considering are the density of certain areas in a binary database, and the support of certain given itemsets. The connection between MaxEnt optimization and the Graphical Models literature will allow us to use results from that community to achieve these goals, such as the Junction Tree algorithm and other techniques for inference and maximum likelihood parameter fitting [23].

An extension similar to this one was made earlier in [11] for randomization approaches to assess data mining results [9]. They introduced different randomization strategies maintaining different properties of a binary dataset besides the row and column marginals, in particular the clustering structure and the frequency of certain itemsets. They suggested this allows data mining to be done in an iterative fashion, updating the randomization model each time a new pattern is reported (and thus becomes part of the prior information). Our framework could accommodate such iterative strategy as well as an alternative to mining pattern sets, as soon as it can handle more complex types of prior information.

Other challenges.

We mentioned earlier that the prior information does not need to be correct for the framework to be useful. However, it would run into problems if the prior information were inconsistent. If this is the case, the method needs to be adapted e.g. by allowing each of the constraints to be vio-

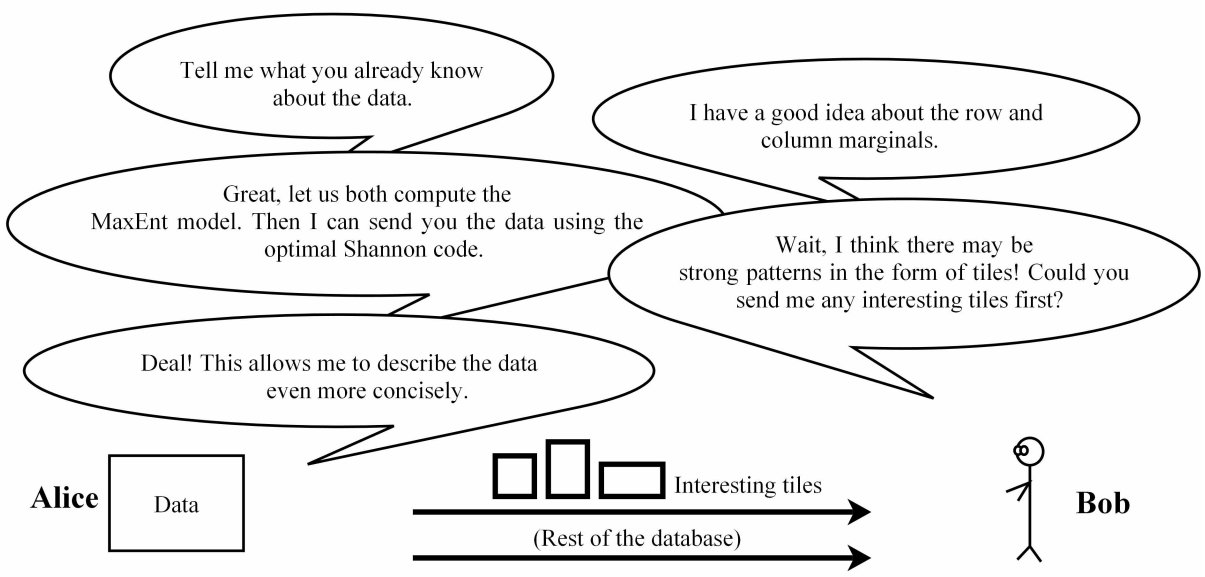


Figure 2: A data mining algorithm implementing our framework can be viewed as an interface moderating the communication between the data miner (Alice) and Bob (the data), trying to help convey the data from Alice to Bob as efficiently as possible. It takes into account what Bob already knows (or thinks to know), as well as the syntactic form of the patterns he believes the data may contain. This figure illustrates that for the running example on mining interesting tiles.

lated by a small amount ε_i. Then, $C \sum_i \varepsilon_i$ can be subtracted from the entropy objective with C akin to a regularization parameter, such that a trade-off is achieved between maximizing the entropy and achieving a good fit to the prior information.

A second important challenge we wish to highlight is the design of efficient algorithms that mine pattern sets as considered in this paper. For the tile mining example, we are currently using a two-step approach, where first all tiles are mined and subsequently they are selected in a greedy way (and thus sorted in the order in which they were selected). It is likely that more efficient algorithms can be devised.

Another challenge is to find out if and how actionability [21] can be incorporated into this scheme. We suspect there may be relations between particular coding schemes for the patterns and certain properties of how they are going to be used, such as the cost of exploiting a pattern or (perhaps equivalently) the profit in exploiting the pattern. However, it is as yet unclear to us if this is the case, or if such connection would be helpful at all.

In this paper, we have chosen to put our framework on statistical foundations. However, it is conceivable that the prior information can be captured in a knowledge base of (possibly probabilistic) logical rules instead. The broad ideas of the framework would stay in place. Doing this would bring the framework closer to the work of [21].

Agreeing on an empirical evaluation strategy.

Finally, a bottleneck we believe this area of research is faced with is the lack of a suitable consensus over how subjective interestingness measures can be assessed empirically, within the scope of a scientific paper. In this paper, we have opted to present some empirical results on a textual data set. The motivation for this is that text is intelligible, certainly if we are familiar with the corpus, and we can assess if the method would provide us with useful insights had we not been familiar with it. However, this risks to raise the wrong impression that the method is supposed to compete with text mining methods (whereas it does not exploit any properties of text at all and any competition would be unfair). Another type of evaluation that is often seen is to use the pattern sets as features for classification. We believe that the relevance of this is limited, as a poor classification accuracy only shows that the features are unrelated to the label, not that they are not interesting. As a result, authors have often resorted to quantitative surrogates such as size of the pattern set and computation times, which are sometimes relevant but usually besides the point when the goal is to design subjective interestingness measures. Therefore, agreeing on an appropriate empirical evaluation strategy is in our opinion critical to progress in this field.

4. CONCLUSIONS

In this paper, we have sketched a possible framework for mining interesting pattern sets. In designing it, our intention was to set it up such that it can operate as an intelligent interface between the data miner and the data, considering both on an equal footing. We designed it as generally as possible, and it is not confined to any particular type of data, pattern, or prior information.

That being said, instantiating the framework for new settings is not always trivial, and issues of computational tractability may arise. This forms perhaps the most important challenge for future research around this framework.

Acknowledgements

This work is supported by the EPSRC grant EP/G056447/1, and by the European Commission through the PASCAL2 Network of Excellence (FP7-216866). KNK is also supported by a University of Bristol Centenary Scholarship.

5. REFERENCES

[1] B. Bringmann and A. Zimmermann. The chosen few: on identifying valuable patterns. In *Proc. of 7th IEEE International Conference on Data Mining (ICDM07)*, 2007.

[2] T. De Bie. Explicit probabilistic models for databases and networks. Technical report, University of Bristol TR 123931, arXiv:0906.5148v1, 2009.

[3] T. De Bie. Maximum entropy models for prior information on rectangular databases. Technical report, University of Bristol TR 125861, submitted, 2010.

[4] L. De Raedt and A. Zimmermann. Constraint-based pattern set mining. In *Proc. of the 2007 SIAM International Conference on Data Mining (SDM08)*, 2007.

[5] A. Gallo, T. De Bie, and N. Cristianini. Mini: Mining informative non-redundant itemsets. In *Proceedings of 2007 KDD*, 2007.

[6] A. Gallo, A. Mammone, T. De Bie, M. Turchi, and N. Cristianini. From frequent itemsets to informative patterns. Technical report, University of Bristol TR 123936, 2009.

[7] F. Geerts, B. Goethals, and T. Mielikäinen. Tiling databases. In *Discovery Science*, 2004.

[8] L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys*, 38(3):9, 2006.

[9] A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. *TKDD*, 1(3), 2007.

[10] B. Goethals, W. Le Page, and M. Mampaey. Mining interesting sets and rules in relational databases. In *Proc. of the 25th ACM Symposium on Applied Computing (ACM SAC)*, pages 996–1000, 2010.

[11] S. Hanhijarvi, M. Ojala, N. Vuokko, K. Puolamäki, N. Tatti, and H. Mannila. Tell me something I don't know: Randomization strategies for iterative data mining. In *Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD09)*, pages 379–388, 2009.

[12] S. Jaroszewicz and D. A. Simovici. Interestingness of frequent itemsets using bayesian networks as background knowledge. In *Proc. of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD04)*, pages 178–186, 2004.

[13] K. Kontonasios and T. De Bie. An information-theoretic approach to finding informative noisy tiles in binary databases. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, 2010.

[14] L. G. Kraft. A device for quantizing, grouping, and coding amplitude modulated pulses. Technical report, Massachusetts Institute of Technology, 1949.

[15] K. Lemmens, T. Dhollander, T. De Bie, P. Monsieurs, K. Engelen, B. Smets, J. Winderickx, B. D. Moor, and K. Marchal. Inferring transcriptional modules from chip-chip, motif and microarray data. *Genome Biology*, 7(R37), 2006.

[16] M. Ojala, G. Garriga, A. Gionis, and H. Mannila. Evaluating query result significance in databases via randomizations. In *Proc. of the 2010 SIAM International Conference on Data Mining (SDM)*, 2010.

[17] M. Ojala, N. Vuokko, A. Kallio, N. Haiminen, and H. Mannila. Randomization of real-valued matrices for assessing the significance of data mining results. In *Proc. of the 2008 SIAM International Conference on Data Mining (SDM08)*, pages 494–505, 2008.

[18] B. Padmanabhan and A. Tuzhilin. A belief-driven method for discovering unexpected patterns. In *Proc. of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD98)*, pages 94Ű–100, 1998.

[19] B. Padmanabhan and A. Tuzhilin. Small is beautiful: discovering the minimal set of unexpected patterns. In *Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD00)*, pages 54Ű–63, 2000.

[20] A. Siebes, J. Vreeken, and M. van Leeuwen. Item sets that compress. In *SIAM Conference on Data Mining*, 2006.

[21] A. Silberschatz and A. Tuzhilin. On subjective measures of interestingness in knowledge discovery. In *Proc. of the 1st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD95)*, pages 275Ű–281, 1995.

[22] N. Tatti. Maximum entropy based significance of itemsets. *Knowledge and Information Systems*, 17(1):57–77, 2008.

[23] M. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[24] M. Zaki and C. Hsiao. CHARM: An efficient algorithm for closed itemsets mining. In *Proc. of the 2nd SIAM ICDM*, 2002.

Point-Distribution Algorithm for Mining Vector-Item Patterns

Anne M. Denton, Jianfei Wu
Department of Computer Science
North Dakota State University
Fargo, ND
{anne.denton,jianfei.wu}@ndsu.edu

Dietmar H. Dorr
Research and Development
Thomson Reuters
St. Paul, MN
dietmar.dorr@thomsonreuters.com

ABSTRACT

An algorithm is presented for finding patterns between sets of continuous attributes and item sets. In contrast to most pattern mining approaches, the algorithm considers multiple continuous attributes as a single vector attribute. This approach results in a separate abstraction level and allows multiple vector attributes to be considered. We show that the pattern mining process can uncover relationships between the vector data and item sets. Filtering according to these patterns can be seen as feature selection at the level of the vector attributes as opposed to individual continuous attributes. In the evaluation, we show that the pattern mining algorithm can more effectively and efficiently achieve this filtering than a direct application of classification algorithms. Patterns are identified by relating item data to the distribution of objects within the vector space that is spanned by the sets of continuous attributes. The Kullback–Leibler divergence provides a quantitative measure that establishes whether the subset defined by an item set differs from the overall distribution of data points. The set-subset relationship of data points, which violates i.i.d assumptions, requires an adaptation of standard algorithms for computing the Kullback–Leibler divergence. The algorithm is evaluated on gene expression data and on a classification example problem that is constructed from time series data.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; I.5.2 [**Pattern Recognition**]: Design Methodology—*Pattern Analysis*

General Terms

Algorithms

Keywords

Pattern mining; Statistical significance; Feature selection

UP'10, July 25th, 2010, Washington, DC, USA.

1. INTRODUCTION

Advances in storage technology have long been driving the need for new data mining techniques. Not only are typical data sets becoming larger. The diversity of available attributes is increasing in many problem domains. This work addresses an aspect of diversity that is often overlooked: Data mining problems commonly involve more than a single set of attributes that fits a vector space model, and may in fact be characterized by multiple separate sets. Those sets provide an additional level of abstraction, and may either derive from separate sets of experiments, or they may have been the result of transformations from different original data types such text, by means of the the bag-of-words model [12], or graphs, through vector space embeddings [24, 23].

The presented work recognizes the cohesiveness of multiple sets of continuous attributes and uses the combinations for further processing. As is commonly done in data mining, machine learning, and statistics, we consider those attributes as dimensions in a vector space. The search for patterns allows focusing on those vector attribute – item set combinations for which a relationship can be shown to exist. The pattern mining can be used as a preprocessing step towards classification with the respective item or item set as class label. Our work differs from conventional approaches in that multiple vector attributes are considered. In contrast to conventional feature selection, our approach determines the usefulness of entire vector attributes, each of which consists of multiple continuous attributes. We focus on data that are already provided in vector format, such as gene expression data, and search for patterns that relate the vector attributes to item attributes, such as those considered in market basket research [1], i.e. attributes that are conceptually binary and represent presence and absence of a particular item with presence typically being less frequent then absence. In the evaluation we show that the pattern mining process finds existing relationships with high accuracy, even when the underlying data has explicitly been chosen to be noisy.

Figure 1 provides an overview of the problem. For simplicity, a one-dimensional setting is chosen, but the concepts of distributions equally apply in higher dimensions. In this figure, the (one-dimensional) vector information determines the horizontal position of an object. Objects are assumed to also have item data. The presence of item 1 is indicated through a filled red circle and the presence of item 2 through a filled green rectangle. Objects that do not show a particular item have an empty circle or rectangle respectively. The

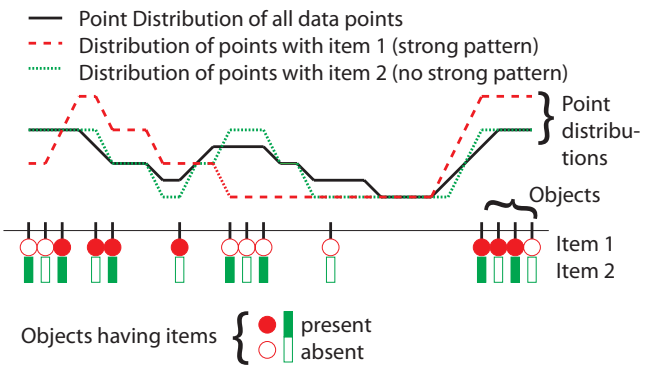


Figure 1: Schematic of a 1-dimensional data set of points together with two example items. Smoothed point distributions for the overall data set as well as for points having the individual items are shown. It can be seen that the distribution of points with item 1 shows two peaks that are much higher than what would be expected based on the overall distribution (strong pattern exists). The distribution of points with item 2 follows the overall distribution much more closely (no strong pattern exists).

distributions are schematically shown above the data points. It can be seen that the distribution of all data points has three peaks, and the distribution of data points with item 2 follows this distribution mostly. The distribution of data points with item 1, in contrast, only shows two peaks, and they are higher than would be expected based on the overall distribution. Hence, a pattern can be observed between the location of objects (vector data) and the presence of item 1 can be observed. No clear pattern is seen between the location information and item 2. The same reasoning can be used for object locations in an arbitrary number of dimensions.

The data mining of vector-item patterns was first introduced in [38]. That work used histograms of density values as basis for evaluating the relationship between any one item and a vector attribute. Significance testing was used to evaluate whether the subset of objects that is defined by an item results in an unexpected density distribution for the vector data. The goal of finding vector-item-patterns does not, however, require the step of summarizing the distribution of objects in a simple histogram. Rather, the density distribution of the complete set of objects can directly be compared with the subset defined by any one of the items.

A well-established measure for the difference between probability distributions is the Kullback–Leibler divergence. The K–L divergence is not symmetric but is rather typically used to compare a true distribution with a model, as is the case in our application. The problem of estimating the K–L divergence from data has been studied extensively [37, 28]. However these techniques typically rely on samples that are independently drawn from the two distributions. The problem of interest in this paper is not suited to these techniques since, by definition, the samples with the item are a subset of all samples. Among other consequences, this means that each data point from the subset distribution is guaranteed to have a neighbor from the superset distribution at distance $d = 0$. One-nearest-neighbor approaches such as [37] can, therefore, not be used.

We use a robust kernel-density-estimation-based technique [27] that makes no assumptions about the relationship between the sampling of the two distributions. Similar techniques are known from kernel-density estimation [27] and kernel-density-based clustering [13, 17]. Instead of a Gaussian kernel, which does not work well in high dimensions [36], we use a uniform kernel, for which the diameter is directly tied to the properties of the data, leaving no free parameters that a user would have to choose.

2. RELATED WORK

The concept of vector item patterns was first introduced in [38] and has been applied to different data sets and similarity measures in [8]. A related algorithm based on pairwise similarity has been shown to assist in determining relevant functional groups in *Escherichia coli* [9].

From the data mining perspective, the problem of relating vector to item data can be considered either from the frequent pattern mining or from the classification perspective. Frequent pattern mining algorithms, which were originally developed for item data alone [1], have been generalized to continuous data [4] and combinations of item and continuous data [33, 29, 5]. These algorithms do not, however, allow considering sets of continuous attributes jointly in the pattern mining process.

From the classification perspective, the concept of a vector-item pattern can be compared with the question of whether classification results are significant [16, 18]: If a class label cannot be predicted based on a set of attributes, we can conclude that those attributes and the class label do not show a pattern. Testing this for a large number of potential class labels, corresponds to the pattern mining problem discussed in this paper, and we use a classification-based technique for comparison purposes. We use a significance test that is based on 2-fold cross validation and treats the confusion matrix [22] as a contingency table.

The vector-item pattern problem is also loosely related to multi-view learning, where the objective is to make predictions based on multiple representations [26] or to use multiple representations for clustering [3] rather than finding which sets of features are most strongly related to which item sets. The problem of relating item sets to multiple continuous attributes is also discussed in the multivariate discretization literature [2]. Note that the presented vector-item pattern mining algorithm does not require discretization and is, therefore, more general.

In bioinformatics applications, the problem of relating continuous attributes, in particular gene expression data to Boolean attributes, such as biological functions, is typically addressed through gene set enrichment analysis, GSEA[35, 34]. In its original form, GSEA takes a single continuous attribute as input and tests whether any one functional category shows comparatively high gene expression overall for genes that have that function or item. A main motivation of this work is to improve statistical significance by considering all genes that have the particular function rather than evaluating the expression of individual genes.

Generalizations have been developed that compare groups of continuous attributes using a hypothesized profile, as discussed in the GSEA documentation, or results of clustering or biclustering [30]. While the latter approach does effec-

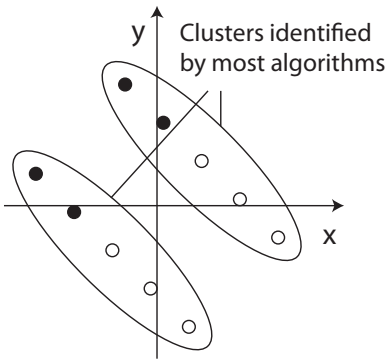


Figure 2: Example of a vector-item pattern that cannot be found by a two-step approach of clustering and enrichment analysis. Two-dimenensional vector information determines the location of objects in the plane. Data points with the items set are show as solid black and others as empty circles.

tively evaluate the relationship between multiple continuous attributes and an item, it suffers from limitations: The initial clustering does not consider any structure given by the item attribute. Fig. 2 illustrates this limitation: The example shows two clear clusters that would be identified by most clustering algorithms. Neither of these clusters shows enrichment for the annotation or item that is represented by the solid circles, since each of them has two out of five members with the item. The genes with the item, nevertheless, show a clear pattern that can be found by vector-item techniques [9]. The pattern in Fig. 2 could also clearly be identified by testing the significance of classification, which we therefore use as comparison approach in the evaluation. Clustering algorithms have also been adapted to use functional or item information [19]. However, such algorithms are not suitable in the context of gene enrichment analysis since they distort the probability of finding enriched annotations.

The Kullback–Leibler, or K–L, divergence and other information-theoretic measures are frequently used in data mining [11, 10]. Often the K–L divergence is beneficial for data that can be modeled through discrete random variables. In the example of text data, each word in the corpus may be considered as one state of a random variable and the frequency of occurrence of the word is directly related to the probability of that state. All words can be modeled through a single random variable. Such an approach is not appropriate for general continuous data, which may not represent a probability distribution. We consider each attribute as a continuous random variable, resulting in a multidimensional or multivariate distribution of data points within the space of attribute values.

3. POINT–DISTRIBUTION ALGORITHM

3.1 Vector and Item Data

The algorithm assumes that multiple (D) continuous attributes are considered to be related based on prior knowledge. Patterns involve all of these attributes $x_i \in \mathbb{R}, 0 \leq i < D$ together as one "vector" attribute $\mathbf{x}$ with domain $\mathrm{dom}(\mathbf{x}) = \mathbb{R}^D$. The set of occurring data points (extant domain) is $V \subset \mathbb{R}^D$. Traditionally, vector space representations are used to describe all independent variables in a particular data mining problem. Our approach differs in that vector attributes are considered as one building block of many.

Item data are conceptually viewed as binary attributes $B^{(i)}, 0 \leq i < M$, that represent the presence of item i, with M distinct items occurring in the database. This view is in accordance with the original formalism used by Agrawal et al. [1], and the related downward closure properties of the support of item sets apply to our problem. However, the Kullback–Leibler divergence measure does not provide any such pruning opportunities.

3.2 Overview

To find patterns among vector and item data we perform the following steps

Normalization: In contrast to histogram-based techniques that depend on the data having a uniform distribution [38], the Kullback–Leibler divergence does not place specific requirements on the normalization. In the evaluation section, we normalize the vector data by subtracting the mean and dividing by the norm of vector, which is typically recommended for both the gene expression and the time series [14] we use. The algorithm does not, however, critically depend on such a normalization.

Determine frequent item sets: Standard downward closure considerations apply to item data.

Determine overall distribution: The distribution based on all data points is calculated, using kernel-smoothing. The kernel-width is chosen such that each the average number of subset points within the kernel hypervolume is one. If this item-support-dependent kernel-width is chosen, the overall distribution has to be recalculated for some representative support values.

Determine subset distribution: Each item set determines a subset of data points that define the subset distribution. The distribution and the overall distribution are sampled based on the data points with the item.

Determine Kullback–Leibler divergence: The Kullback-Leibler divergence is calculated for the subset distribution with respect to the overall distribution.

Repeat process for multiple vector attributes: The process can be repeated for different vector attributes.

3.3 Kullback–Leibler Divergence

The Kullback–Leibler, or K–L, divergence is a measure of the similarity of probability distributions. It can be written as

$$D_{\mathrm{KL}}^{(d)}(P||Q) = \int_{-\infty}^{\infty} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} \, \mathrm{d}x_1, \ldots, \mathrm{d}x_d \qquad (1)$$

where q is the reference distribution, which in our case is the full distribution of all objects, and p is the distribution of objects that have the item of interest. Both distributions are assumed to be defined over a d-dimensional vector space. Using the law of large numbers this expression can be estimated as follows

$$D_{\mathrm{KL}}^{(d)}(P||Q) = \frac{1}{n} \sum_{i=1}^{n} \log \frac{p(\mathbf{x}^{(i)})}{q(\mathbf{x}^{(i)})} \qquad (2)$$

where the $\mathbf{x}^{(i)}$ are samples drawn according to the distribution $p(\mathbf{x})$. The samples that have the item are assumed to be independent and identically distributed, i.i.d..

Since we do not know the probability density from which the observed points are sampled, we have to estimate it. The estimation of a probability density function from a distribution of data points goes back to Parzen [27] and is commonly done by kernel-smoothing. We use a uniform kernel since Gaussian kernels are known to be problematic in high dimensions [36]. The similarity is determined based on whether the product exceeds a threshold. For data that are normalized, such that they lie on a hypersphere of radius one, this results in a uniform kernel that is equivalent one defined based on Euclidean distance. A threshold on a Euclidean distance t_E can be converted to a product threshold t given the normalization

$$\begin{aligned} t_E &= \sqrt{\left(\frac{\mathbf{x}}{|\mathbf{x}|} - \frac{\mathbf{y}}{|\mathbf{y}|}\right)^2} = \sqrt{2\left(1 - \frac{\mathbf{x}}{|\mathbf{x}|} \cdot \frac{\mathbf{y}}{|\mathbf{y}|}\right)} \\ &= \sqrt{2(1-t)} \end{aligned} \tag{3}$$

We use the product threshold because the complexity of calculating products is lower than of calculating Euclidean distances. A kernel-density estimator for n data points $x_i, i = 1, \ldots, n$ in a d-dimensional space is given by

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \tag{4}$$

where the kernel function $K(\mathbf{x})$ is normalized

$$\int_{\mathbf{R}^d} K(\mathbf{x}) \mathrm{d}^d x = 1. \tag{5}$$

A uniform kernel is used for this purpose since Gaussian kernels are known to be problematic in high dimensions [36]:

$$K(\mathbf{x}) = \frac{\theta(1 - |\mathbf{x}|)}{2^d V_d} \tag{6}$$

$$\hat{f}(\mathbf{x}^{(\Omega)}) = \frac{1}{N} \sum_{i=1}^{N} K\left(\mathbf{x}^{(\Omega)}, \mathbf{x}_i^{(\Omega)}\right) \tag{7}$$

The superscript $^{(\Omega)}$ indicates that all vectors are normalized to $|\mathbf{x}|$=1.

$$K\left(\mathbf{x}^{(\Omega)}, \mathbf{x}_i^{(\Omega)}\right) = \frac{\theta\left(t - \mathbf{x}^{(\Omega)} \cdot \mathbf{x}_i^{(\Omega)}\right)}{A_d} \tag{8}$$

where θ is the Heaviside step function, and A_d is the section of the d-hypersphere surface, for which the step function is one

$$\int_{\Omega} K\left(\mathbf{x}^{(\Omega)}, \mathbf{x}_i^{(\Omega)}\right) \mathrm{d}\mathbf{x}^{(\Omega)} = A_d. \tag{9}$$

ensuring that the kernel function is appropriately normalized.

3.4 Choice of Kernel Width

We chose a kernel width such that one data point is expected within the hyper-volume that is covered by the kernel, see Fig. 3. The rationale for this choice is that once more than one data point is expected within the kernel volume, local properties are lost. A smaller kernel volume, on the other hand, would result in fluctuations of the probability density even for data points that are approximately

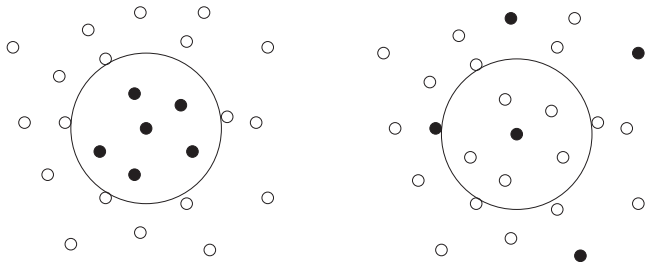


Figure 3: Kernel-density-based probability distribution function with the kernel width chosen such that one data point with item set is expected per kernel volume. The density is based on the number of points within the fixed kernel volume.

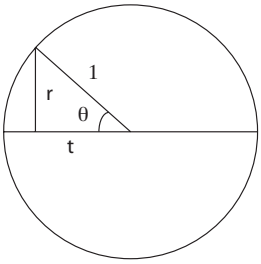


Figure 4: Schematic to illustrate the integration of the cap of a hypercube.

equidistant. One may still be concerned that even a random distribution of data points can show large variations in $p(\mathbf{x})$ if the expected number of points per kernel volume is one. We account for random contributions to the K–L divergence by comparing with randomly created subsets of the data. Nevertheless, one may argue that a choice of kernel-width that removes some of the random fluctuations in $p(\mathbf{x})$ might be beneficial. Our study in Sect. 4.3 shows that there is no empirical basis for this arguments and that, in fact, a kernel-width corresponding to one expected subset point per kernel volume does consistently lead to the strongest results. A likely explanation is that fluctuations in random distributions are expected to be relevant at all length scales. Choosing a larger kernel width does not fundamentally resolve any of the randomness.

The number of available samples for $p(\mathbf{x})$ and $q(\mathbf{x})$ typically differ widely, since $p(\mathbf{x})$ only has as many available samples as there are objects with the item. This number is normally much smaller than the total number of objects. We use the same kernel function for $q(\mathbf{x})$ as for $p(\mathbf{x})$ to ensure that those data points that are considered for $q(\mathbf{x})$ include the samples considered for $p(\mathbf{x})$. Hence, there are typically many more than one data point within the kernel volume for $q(\mathbf{x})$.

Following these assumptions, the threshold parameter t is to be calculated such that the expected number of data points is $f = 1/N_i$, where N_i is the support of the item set in question. Fig. 4 shows the setup: We calculate the surface of the hypersphere cap in d dimensions by integrating over contributions for those values of θ that are smaller than the threshold. For each value of θ, the surface of a hypersphere in $d-1$ dimension with radius r contributes to the integral. For example, if the total density were three the points of

constant θ lie on a circle of radius r. Since the circle itself is a 1-dimensional structure it is denoted as S_1. The surface of the cap can be calculated by integrating over the circles for all values of θ that satisfy the t-threshold. In d dimensions we get

$$S_{d-1}^{(\mathrm{cap})} = \int_0^{\theta_t} S_{d-2}\, r^{d-2}\, \mathrm{d}\theta = \int (d-1)\, C_{d-1}\, r^{d-2}\, \mathrm{d}\theta \quad (10)$$

where S_{d-2} is the surface of a hypersphere of radius 1 in $d-1$ dimensions. C_{d-1} is the volume of a unit hypersphere in $d-1$ dimensions. Using

$$\begin{aligned} r &= \sin\theta \\ \frac{\mathrm{d}r}{\mathrm{d}\theta} &= \cos\theta \\ \mathrm{d}r &= \sqrt{1-r^2}\, \mathrm{d}\theta \end{aligned} \quad (11)$$

we can rewrite $S_d^{(\mathrm{cap})}$ as follows assuming threshold $t > 0$:

$$S_{d-1}^{(\mathrm{cap})} = d\, C_{d-1} \int_0^{\sqrt{1-t^2}} \frac{r^{d-2}}{\sqrt{1-r^2}}\, \mathrm{d}r \quad (12)$$

This integral can be performed analytically in Matlab. Calculating the hypershere volume C_{d-1} can be avoided since only the ratio $S_{d-1}^{(\mathrm{cap})}/S_{d-1}$ is needed, which can be evaluated by integrating equation (12) up to $t = 0$. For "support" being the absolute item set support we get

$$\frac{1}{\mathrm{support}} = \frac{\int_0^{\sqrt{1-thresh^2}} \frac{r^{d-2}}{\sqrt{1-r^2}}\, \mathrm{d}r}{2\int_0^1 \frac{r^{d-2}}{\sqrt{1-r^2}} \mathrm{d}r} \quad (13)$$

In practice, are interested in *thresh* as a function of "support". However, eq. (13) cannot be inverted analytically. We use an interval-halving algorithm for inverting it numerically.

3.5 Algorithm Outline

The complete algorithm is summarized in Algorithm 1. Array-based notation similar to what is used in Matlab is assumed. The algorithm takes the data points as input. It is assumed that frequent item sets have been determined previously. For simplicity, we also assume that the presence of items and item sets is provided in a binary array. It would be straightforward to modify the algorithm to use lists of items, but such an implementation is less concise in Matlab. It is also assumed that a cutoff of the K–L divergence has been determined from random subsets of the data. We use the mean plus twice the standard deviation over a set of random filters as cutoff.

Next, an array of similarity values is determined. In the evaluation, data set sizes were small enough to keep this array in memory, but the algorithm could easily be rewritten, to avoid storing the full array. Lines 5-8 in Algorithm 1 depend somewhat on the item set support and are, therefore, performed for representative support values. It will be shown, that the representative support can differ from the actual support by a factor of two without substantially affecting results.

The similarity threshold is then determined, given the item set support, using eq. (13) and the interval halving process motivated there. The binary matrix of those point combinations that satisfy the similarity threshold (*sim*) is determined. Summing over the rows of this matrix (line 8) gives the overall density. The subset density for the given item set is determined by selecting only those rows that have the particular item set (line 12). The K–L divergence for the respective item set is then calculated.

Algorithm 1: Point–Distribution Algorithm

```
   Data: pts;                    /* normalized data points */
   Data: itemSetFilters;         /* binary array */
   Data: klRandCutoff ;          /* mean + 2 std of KL of
         random items */
   Result: highKLItems;          /* items with K-L
           divergence more than random */
1  simVal = getSimilarityMatrix(pts);
2  supportVals = sum(itemSetFilters);
3  repSupportVals = findReps(supportVals);
4  foreach repSup ∈ repSupportVals do
5      filters = findFilters(itemSetFilters, repSup);
6      thresh = invCapRatio(dim(pts), 1/repSup);
7      sim = simVal > thresh;
8      densAll = sum(sim)/count(sim);
9      foreach f ∈ filters do
10         i = indexOf(f);
11         subsetSim = sim(i, :);
12         densI = sum(subsetSim)/count(subsetSim);
13         kl(i) = sum(log(densI./densAll));
14 highKLItems = kl > klRandCutoff;
15 return highKLItems
```

3.6 Comparison with Classification-based Algorithm

While the objective of this paper is not directly classification, it is nevertheless possible to use classification algorithms to derive corresponding results. To do so, items are considered to be class labels. If the item can be successfully predicted based on the vector data using classification then it is fair to assume that there is a relationship between the vector data and the particular item. This process has to be repeated for each item attribute. We consider classification to be successful if the confusion matrix, treated as a contingency table, shows a significant relationship between actual and predicted class labels on the test set. We use 2-fold cross-validation to derive the confusion matrix and determine significance based on χ^2 testing. Since elements within the confusion matrix may be small, the Yates correction [39] is applied.

Notice that classification-based measures, such as accuracy, are also used in Section 4.2. In that section a data set is constructed such that some items are expected to show a significant relationship to the vector data, while others don't. Such an analysis is performed both for the classification-based comparison and for the point-distribution algorithm and should not be confused with the classification-based algorithm discussed in this section. For the classification-based comparison algorithm, accuracy is not directly of interest, since the entire confusion matrix is necessary to determine whether the classification is successful.

3.7 Comparison with Previous Approaches

The histogram-based approaches of [38] and [8] that are used as comparison algorithms have the same objective as

Table 1: Results for Gene Expression Data

All	Alpha	Cdc15	Cdc28	Elu	
259	114	117	160	166	All
	119	67	85	79	Alpha
0		134	86	72	Cdc15
7E-15	**3E-12**		173	107	Cdc28
0	**2E-16**	**7E-12**		198	Elu
0	**1E-7**	**0.027**	**5E-8**		

this paper: to identify item sets that show a significant pattern with respect to a vector attribute. In those papers the subset distributions are summarized using histograms. Those histograms are compared with expected histograms that are either derived through resampling or through a theoretical model. The comparison runs in this paper all use resampling, which results in higher accuracy but slower speed. In [38] similarity is evaluated using a subspace-based similarity measure, and [8] also presents results using a product similarity measure. Results for both similarity measures are presented in the comparison.

4. IMPLEMENTATION AND EVALUATION

The algorithm was implemented in MATLAB. A bitvector representation was used for items because of the conciseness of array-based implementations MATLAB. We did not experience memory constraints for any of the data sets in the evaluation. Performance results are based on running the code on a Mac power book with a 2.6 GHz Intel Core 2 Duo Processor with 4GB 667 MHz SDRAM memory in a Vista shell using VMware Fusion. For the classification-based comparison the Classification Tree classifier within the MATLAB Statistics Toolbox is used and significance is evaluated based on 2-fold cross-validation. A χ^2 significance test with Yates correction is used on the confusion matrix as contingency table. A p-value smaller than 0.05 is considered to indicate a significant result.

4.1 Evaluation on Gene Expression Data

The algorithm is first evaluated on gene expression data sets from cell cycle experiments on yeast [32], which are available for download at [31]. Results from four different types of experiments are available, each of which includes measurements at between 14 and 24 time points. The results come from different types of experiments, but all of them show cell-cycle-related time-dependent gene expression. We only consider genes, for which all four experiments are reported which leaves 5878 of the total of 6178 genes. Item data were extracted from Interpro database that contains information on many types of of sequence signatures including protein domains and motifs [25]. For simplicity, we refer to all sequence signatures as domains. Yeast domain information was downloaded from [6]. We only consider domains with at least 10 instances, leaving 432 domains.

The goal of the analysis in this section is to identify protein domains, that are significantly related to cell cycle activity. Conventionally, biologists would either look at the lists of differentially expressed genes, and try to find domains or functions that occur particularly often within those sets. Alternatively, they might use clustering or biclustering techniques and then test those clusters for enrichment. Both types of reasoning correspond to two-step approaches,

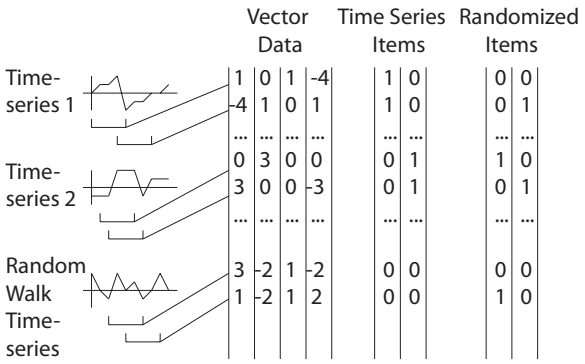


Figure 5: Schematic showing how the time series data set is designed. Each window (after preprocessing) corresponds to one vector. Items that are constructed by assigning a label to those vectors that come from one particular time series are expected to be significant (time series items). Items that constructed by randomly assigning labels (randomized items) are not expected to be significant.

in which the expression data are first analyzed with traditional techniques and relationships with domain or functional data are evaluated based on the already completed evaluation of the first step. Our approach, in contrast, directly searches for patterns between the expression data and domain or functional information respectively.

Since we do not know, which domains are expected to be significantly related to the cell cycle, we limit the discussion in this section to a comparison among the four different data sets, as well as the combination of all four. For a quantitative evaluation against an independent standard we refer to the next section on time series data. Table 1 shows, in its diagonal, the total number of domains that are considered significant according to the experiments. The "All" entry refers to the combination of all four experiments (73 columns) and "Alpha" (18 columns), "Cdc15" (24 columns), "Cdc28" (17 columns), and "Elu" (14 columns) are used as in the description of the original experiments. The overlap in results is shown in the upper right corner of the table. The lower left corner shows whether that overlap is significant according a χ^2 significance test using Yates correction. The largest p-value is 0.02, which indicates that all values for overlap are significant. Several of the entries have such low p-values that Matlab reported 0. The p-values are more consistently low than those reported using histogram-based techniques [38].

The consistency between results from different experiments suggests that the results do indeed represent patterns that are biologically relevant. The comparison with the full set of all experiments is primarily included to test whether the algorithm is capable of extracting meaningful results for high-dimensional data. It should be expected that results are consistent between a set of experiments and one of its subset. This comparison leads to p-values below the Matlab resolution in all cases, confirming that algorithm works well for the 73 dimensions of the full data set.

4.2 Quantitative Evaluation on Time Series Data

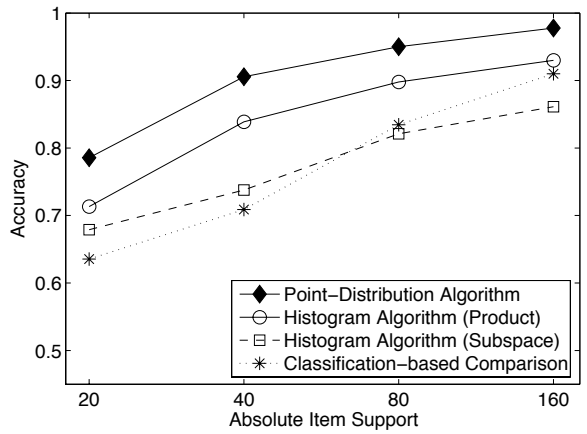


Figure 6: Accuracy depending on the absolute item support, $Npts$. Two batches of $9*Npts$ random points are added in each setting, i.e. the relative item support is 0.037 throughout.

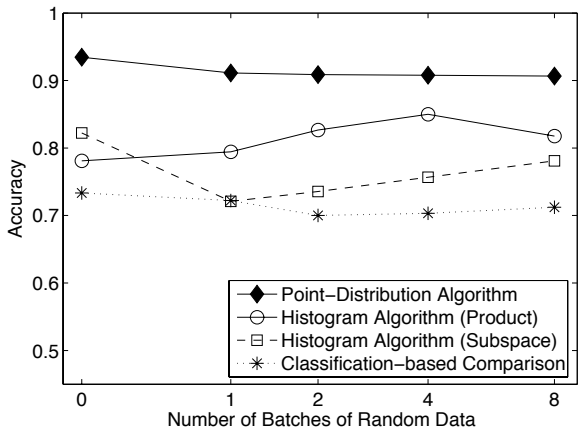


Figure 7: Accuracy for $Npts = 40$ time series subsequences. Batches of $9 * Npts = 360$ random points are added. The reported results range from 360 to 6120 points in total.

Secondly, the algorithm is tested on time series subsequence data from the same nine time series as used in [7]. Subsequences are extracted from the *buoy sensor*, *balloon*, *glass furnace*, *steamgen*, *speech*, *earth quake*, *ocean*, and *darwin* of the UCR time series repository [20]. Descriptions of these data sets are distributed with the data. One time series (*ecg*) was collected independently: The Ecg series (MIT-BIH Arrhythmia Database: mitdb100) originates from PhysioBank [15]. The same preprocessing steps were done as in [7]: The *buoy sensor* series was compressed by averaging over 4 consecutive values and the *ecg* series by averaging over 20 consecutive values.

Figure 5 shows the design of the time series data set. Nine items are constructed, such that they can be expected to show a vector-item pattern with the subsequence data. Each one of these items is associated with one of the original time series. The binary item variable is constructed such that it is 1 for the subsequences of its associated time series and 0 for all other subsequences, including the random walk subsequences. We will refer to these items as time series items. Nine other items, which we call randomized items, are constructed through random selection of subsequences. Using this setup we do a quantitative evaluation of the algorithms using classification-style measures, in which time series items are the positive examples and randomized items are the negative examples. It is important to understand that although this evaluation uses classification concepts there is no training involved. Time series subsequences were chosen since time series subsequence data are notorious for being hard to cluster meaningfully [21]. The averaging over 9 series is furthermore expected to make the results generalizable.

Subsequences are extracted using a sliding window with $w = 17$, and differences between successive data points are taken, resulting in a 16-dimensional vector space, i.e. 2^4 dimensions. $Npts$ subsequences that are randomly chosen from the first 1000 windows of each time series are used from each time series, i.e. the absolute support of all items is $Npts$. In total $Nsets*Npts$ subsequences are non-random, where $Nsets = 9$ is the number of time series. These are combined with $Nrand*Nsets*Npts$ random subsequences, where $Nrand$ is a parameter that is varied from 0 to 8.

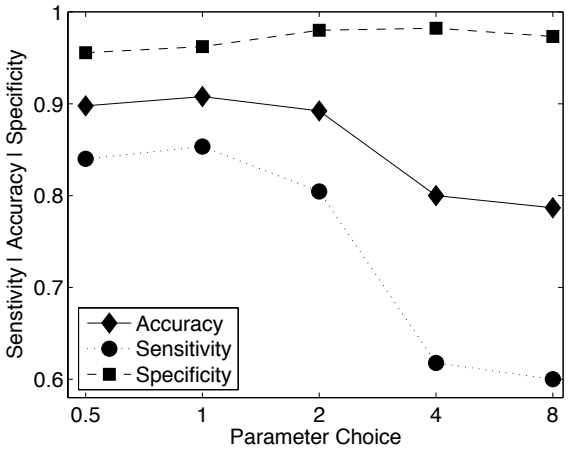


Figure 8: Effectiveness as a function of the choice of a parameter that is defined as the expected number of data points having the item set per kernel volume.

$Nsets = 9$ positive items are constructed from the original time series and $Nsets$ are constructed assigned by randomly selecting sequences. Averages over 50 such runs are reported. Unless otherwise stated, the product algorithm choses a threshold of one expected data point with item per kernel hypervolume. In general, accuracy is reported, although Fig. 8 also shows sensitivity and specificity. Data sets are constructed such that the number of positive and negative items is equal, thereby avoiding problems that are encountered when the class label is skewed.

4.3 Effectiveness

First we vary $Npts$ and keep $Nrand = 2$, i.e. two batches of $Nsets * Npts$ points are added in each case. That means that the absolute support is varied, while the relative support is kept constant ($1/27 \sim 0.037$). Fig. 6 shows the accuracy in comparison with algorithms described in [38] and [8] and the classification-based comparison approach discussed in Section 3.6. It can be seen that the K–L-based algorithm

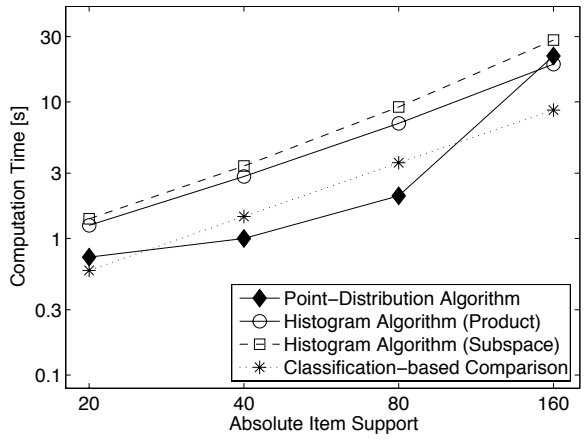


Figure 9: Computation time corresponding to Fig. 6. The total number of data points varies from 540 to 4320 within the shown range.

outperforms all comparison algorithms. Next we keep the item support fixed at $Npts = 40$ and vary $Nrand$. Fig. 7 shows that the K–L-based approach outperforms all comparison algorithms.

Fig. 8 shows the dependence of accuracy, sensitivity, and specificity on the choice of threshold for $Npts = 40$ and $nRand = 2$. The parameter choice indicates how many points with item set are to be expected per kernel volume. The plot confirms that the accuracy is highest for one expected point and falls off quickly for smaller choices. For larger choices it decreases more slowly. Fig. 8 also reports sensitivity and specificity separately, showing that it is the sensitivity that most strongly depends on parameter choice, while the specificity remains stable over the tested parameter range.

4.4 Efficiency

The design of the algorithm was, so far guided by the goal of achieving accuracy more then efficiency. For the data set sizes used in the evaluation, the execution time is nevertheless comparable with histogram-based approaches as well as the classification-based comparison using the Matlab tree-classifier, as can be seen from Fig. 9. Notice that for the classification-based comparison, a fast tree-based classifier was used.

5. CONCLUSIONS

We have introduced an algorithm to mine vector-item patterns using Kullback–Leibler distributions of kernel densities. We have demonstrated that the resulting algorithm is far more effective than previous algorithms, which summarized point distributions first and then compared the histogram summaries. We have also shown that the approach is more effective at determining vector-item relationships than directly determining the significance of classification. We have discussed the choice of the only parameter in the algorithm, in particular the width of the kernel function, and validated the choice empirically. We have also shown that in a practical application to gene expression data, our resulting patterns are consistent among biologically distinct experiments.

6. REFERENCES

[1] R. Agrawal, T. Imielinski, and A.N. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[2] S. Bay. Multivariate discretization for set mining. *Knowledge and Information Systems*, 3:491–512, 2001.

[3] S. Bickel and T. Scheffer. Multi-view clustering. In *Proc. Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 19–26, 2004.

[4] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: generalizing association rules to correlations. In *SIGMOD '97: Proc. of the 1997 ACM SIGMOD Int'l Conf. on Management of Data*, pages 265–276, New York, NY, USA, 1997. ACM Press.

[5] R. Chiang, C.E. Huang Cencil, and E.-P. Lim. Linear correlation discovery in databases: a data mining approach. *Data and Knowledge Engineering*, 53:311–337, 2005.

[6] Saccharomyces Genome Database. InterProScan results using s. cerevisiae protein sequences ftp://genome-ftp.stanford.edu/pub/yeast/sequence_similarity/domains/domains.tab.

[7] A. Denton. Kernel-density-based clustering of time series subsequences using a continuous random-walk noise model. In *Proc. 5th IEEE Int'l Conference on Data Mining (ICDM'05),*, pages 122–129, Houston, TX, 2005.

[8] A.M. Denton and J. Wu. Data mining of vector-item patterns using neighborhood histograms.

[9] A.M. Denton, J. Wu, M.K. Townsend, and B.M. Prüß. Relating gene expression data on two-component systems to functional annotations in *Escherichia coli*. *BMC Bioinformatics*, 9:294, 2008.

[10] I. S. Dhillon, S. Mallela, and D.S. Modha. Information-theoretic co-clustering. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98, 2003.

[11] I.S. Dhillon, S. Mallela, and R. Kumar. A divisive information theoretic feature clustering algorithm for text classification. *J. Mach. Learn. Res.*, 3:1265–1287, 2003.

[12] R. Feldman and J. Sanger. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2006.

[13] P. Foster. Exploring multivariate data using directions of high density. *Statistics and Computing*, 8:347 – 355, 1998.

[14] M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani. Mining the stock market (extended abstract): which measure is best? In *Proc. of the 6th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, pages 487–496, Boston, MA, 2000.

[15] A.L. Goldberger, L.A.N. Amaral, L. Glass, et al. PhysioBank, PhysioToolkit, and PhysioNet. *Circulation*, 101(23):e215–e220, 2000. Circulation Electronic Pages: [http://circ.ahajournals.org/cgi/content/full/101/23/e215].

[16] P. Golland, F. Liang, S. Mukherjee, and

D. Panchenko. Permutation tests for classification. In *Proc. COLT: Annual Conference on Learning Theory*, volume LNCS 3559, pages 501–515, 2005.

[17] A. Hinneburg and D.A. Keim. A general approach to clustering in large databases with noise. *Knowlege and Information Systems*, 5(4):387–415, 2003.

[18] T. Hsing, S. Attoor, and E. Dougherty. Relation between permutation-test p values and classifier error estimates. *Machine Learning*, 52(1-2):11–30, 2003.

[19] S. Kaski, J. Sinkkonen, and J. Nikkilä. Clustering gene expression data by mutual information with gene function. In *Proc. Int'l Conf. on Artificial Neural Networks (ICANN)*, pages 81–86, 2001.

[20] E. Keogh and T. Folias. The UCR time series data mining archive, accessed 2003. [http://www.cs.ucr.edu/~eamonn/TSDMA/index.html].

[21] E.J. Keogh, J. Lin, and W. Truppel. Clustering of time series subsequences is meaningless: implications for previous and future research. In *Proc. IEEE Int'l Conf. on Data Mining*, pages 115–122, Melbourne, FL, 2003.

[22] R. Kohavi and F. Provost. Special issue on applications of machine learning and the knowledge discovery process. *Machine Learning*, 30:271–274, 1998.

[23] N. Linial, A. Magen, and M. E. Saks. Low distortion euclidean embeddings of trees. *Israel Journal of Mathematics*, 106:339–348, 1998.

[24] B. Luo, R.C. Wilson, and E.R. Hancock. Spectral embedding of graphs. *Pattern Recognition*, 36:2213–2223, 2003.

[25] N.J. Mulder, R. Apweiler, and T.K. Attwood. New developments in the InterPro database. *Nucleic Acids Research*, 35:D224–228, 2007.

[26] I. Muslea, S. Minton, and C. A. Knoblock. Active learning with multiple views. *Journal of Artificial Intelligence Research*, 27:203–233, 2006.

[27] E. Parzen. On estimation of a probability density function and mode. *Ann. Math. Stat.*, 33:1065–1076, 1962.

[28] F. Perez-Cruz. Kullback-leibler divergence estimation of continuous distributions. In *NIPS '07 Workshop on Representations and Inference on Probability Distributions*, 2007.

[29] R. Rastogi and K. Shim. Mining optimized support rules for numeric attributes. *Information Systems*, 26(6):425–444, 2001.

[30] R. Shamir, A. Maron-Katz, A. Tanay, C. Linhart, I. Steinfeld, R. Sharan, Y. Shiloh, and R. Elkon. Expander-an integrative program suite for microarray data analysis. *BMC Bioinformatics*, 6:232, 2005.

[31] P.T. Spellman. Yeast cell cycle analysis project, http://cellcycle-www.stanford.edu/, accessed 2007.

[32] P.T. Spellman, G. Sherlock, M.Q. Zhang, et al. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297, 1998.

[33] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. 1996 ACM SIGMOD Int'l Conf. on Management of Data*, pages 1–12, Montreal, Quebec, Canada, 4–6 1996.

[34] A. Subramanian, H. Kuehn, J. Gould, P. Tamayo, and J.P. Mesirov. Gsea-p: a desktop application for gene set enrichment analysis. *Bioinformatics*, 23:3251–3253, 2007.

[35] A. Subramanian, P. Tamayo, V.K. Mootha, et al. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proc. Natl. Acad. Sci. USA*, 102:15545–15550, 2005.

[36] M. Verleysen and D. François. The curse of dimensionality in data mining and time series prediction. *Lecture Notes in Computer Science*, 3512, 2005.

[37] Q. Wang, S.R.Kulkarni, and S. Verdu. A nearest-neighbor approach to estimating divergence between continuous random vectors. In *2006 IEEE International Symposium on Information Theory*, pages 242–246, 2006.

[38] J. Wu and A. Denton. Mining vector-item patterns for annotating protein domains. In *Proc. of the Workshop on Mining Multiple Information in conj. with the ACM SIGKDD Int'l Conf. on Data Mining (KDD)*, San Jose, Aug 2007.

[39] F. Yates. Contingency table involving small numbers and the χ^2 test. *Journal of the Royal Statistical Society (Supplement)*, 1:217–235, 1934.

Margin-Closed Frequent Sequential Pattern Mining

Dmitriy Fradkin
Siemens Corporate Research
755 College Road East
Princeton, NJ 08540
dmitriy.fradkin@siemens.com

Fabian Moerchen
Siemens Corporate Research
755 College Road East
Princeton, NJ 08540
fabian.moerchen@siemens.com

ABSTRACT

We present a new approach to mining sequential patterns that significantly reduces the number of patterns reported, favoring longer patterns and suppressing shorter patterns with similar frequencies. This is achieved by mining only margin-closed patterns whose support differs by more than some margin from any extension. Our approach extends the efficient BIDE algorithm to enforce the margin constraint. The set of margin-closed patterns can be significantly smaller than a set of just closed patterns while retaining the most important information about the dataset. This is shown by an extensive empirical evaluation on six real life databases.

1. INTRODUCTION

Temporal data mining exploits temporal information in data sources in the context of data mining tasks such as clustering or classification. Many scientific and business data sources are dynamic and thus promising candidates for application of temporal mining methods. For an overview of methods to mine time series, sequence, and streaming data see [17, 14].

One particular type of temporal data are sequences of (sets of) discrete items associated with time stamps, for example histories of transactions of customers in an online shop or log messages emitted by machines or telecommunication equipment during operation. A common task is to mine for local regularities in this data by looking for sequential patterns [2] that represent a sequence of itemsets possibly with gaps in the observation sequences.

It is well know that frequent itemset mining suffers from a combinatorial explosion of results when lowering the minimum support threshold. When mining sequential patterns, i.e., sequences of itemsets on sequential databases, this effect typically becomes even stronger. A lossless way of reducing the number of reported patterns that favors longer, thus more interpretable patterns, is mining of closed patterns. Only patterns that cannot be extended with additional items without lowering their support are reported. A straightforward extension of closed itemset mining are margin-closed itemsets, also known as δ-tolerance itemsets [12]. A margin closed pattern cannot be extended by additional items without lowering the support significantly, as determined by a relative or absolute threshold.

In this work we present an efficient algorithm for mining of margin-closed sequential patterns. The well known BIDE (BI-Directional Extension checking) [39] algorithm is extended to enforce the margin-closed constraints. We show that on real life data the number of reported patterns can be greatly reduced even with moderate margin thresholds. Using a classifier we show that the suppressed (almost redundant) patterns were not of great importance for a specific data mining task.

Related work, mostly in the area of itemset and sequence mining, is described in Section 2. The technical part describes preliminary definitions (Section 3.1), the original BIDE algorithm (Section 3.2) and the proposed extension BIDE-Margin (Section 3.3). Section 4 contains results of evaluation on real life data. Conclusion is presented in Section 5.

2. RELATED WORK

Many publications explored the question of reducing the number of patterns within a general pattern mining framework [20, 6]. In the sections below we discuss methods focused on itemset and sequential mining as being most relevant to our work.

2.1 Itemset mining

Researchers have proposed many solutions to reduce the number of itemset patterns depending on the context in which the patterns are used, for example, condensed representations [8], constrained itemsets [33] and combinations thereof [4, 13], and compression [38, 37]. For association rule generation, closed itemsets [31, 5] are commonly used to avoid redundant rules [43] favoring longer patterns to generate specific rules. For frequency queries non-derivable itemsets [7] provide a compact lossless representation favoring shorter patterns to keep the summary small.

Margin-closed itemsets have been previously proposed by the authors for exploratory knowledge discovery tasks in the context of temporal data mining [25, 27] and independently as δ-tolerance itemsets for frequency estimation in [12]. Margin-closed patterns are a specialization of closed itemsets with a constraint to limit the redundancy among reported patterns. An itemset is closed if no superset with the same frequency exists. An itemset is margin-closed if no superset with almost the same frequency exists, where 'almost' is defined by a threshold α on the relative (or absolute) difference of the frequencies. The threshold ensures a frequency *margin* among the reported patterns. An efficient algorithm for mining margin-closed itemsets, extending the well-know DCI_Closed algorithm [21], has been proposed in [24].

A related line of work is motivated by the fact that transaction data is often noisy. The strict definition of support, requiring all

UP'10, July 25th, 2010, Washington, DC, USA.

items of an itemset to be present in a transaction, is relaxed, see [16] and references therein. These approaches can reveal important structures in noisy data that might otherwise get lost in a huge amount of fragmented patterns. One needs to be aware though that such approaches report approximate support values and possibly list itemsets that are not observed as such in the collection at all [1] or with much smaller support.

2.2 Sequential mining

An overview of algorithms for sequential pattern mining is given in [44]. Our approach extends the BIDE algorithm [39] that uses a smart search space pruning and does not require the patterns found to remain in memory until the algorithm terminates.

Motivated by approaches that have worked on itemsets, research on reduction of the output of sequential pattern mining algorithms includes compression of the mining result in a post-processing step [10, 41], a condensed representation to evaluate sequential association rules [35], and approximate patterns [45] under the Hamming distance.

These approaches are different from the one presented here in at least one of the following ways:

- The margin constraint favors longer patterns, whereas condensed representation focus at reconstruction of frequencies for patterns not reported or compression ratio of the complete pattern set.
- Patterns observed exactly as is, with exact frequencies, are reported, whereas approximate patterns represent observations that may differ (slightly).
- The pruning is integrated in the mining algorithm whereas compression is a postprocessing of the results after mining.

The presented approach therefore has merits in particular if the patterns are used in a context that requires interpretation, as opposed to automated post processing with other algorithms. Longer patterns are more interpretable because they offer more context to the analyst. While approximations that tolerate errors may be more robust, they may report approximate support values and possibly list itemsets that are not observed as such in the collection at all or with much smaller support. This might be misleading in exploratory applications.

A generalization of sequential patterns are partial orders [9]. Instead of requiring a full ordering of the itemsets in a pattern some order relation may be unspecified. This is typically represented by a directed graph of itemsets. In [34] it is shown that closed partial orders are also a generalization of Episodes [22] that are restricted to combinations of fully ordered and completely unordered patterns.

In [9] closed partial orders are mined by grouping of sequential patterns and generating directed graphs. In [26] it is shown that the grouping corresponds to an instance of the closed itemset mining problem. [36] describes an algorithm to mine arbitrary (not necessarily closed) groups of sequential patterns. Experiments on real life data in [29] show that the grouping can both reduce or increase the number of patterns found, depending on the dataset.

3. MINING MARGIN-CLOSED SEQUENTIAL PATTERNS

3.1 Preliminaries

DEFINITION 3.1. *An event sequence over a set of events Σ is a sequence of pairs (t_i, s_i) of event sets $s_i \subseteq \Sigma\ \forall i = 1, \ldots, n$ and time stamps $t_i \in \Re^+$. The ordering is based on time, i.e. $\forall i < j : t_i \leq t_j$. The length of the event sequence is n.*

For most of the discussion in our work (and in much of sequential pattern mining literature) the exact values of the time stamps are not as important as the ordering that they impose. We will therefore omit timestamps from discussion and will treat event sequences as just an ordered set of event sets $S = \{s_i\}$.

DEFINITION 3.2. *A sequence database, SDB, of size N is a collection of event sequences P_i, $i = 1, \ldots, N$.*

We now need to introduce a basic definition from order theory.

DEFINITION 3.3. *A partial order is a binary relation $\prec$ over a set S which is reflexive, antisymmetric, and transitive, i.e., for all $a, b, c \in P$, we have that:*

- *$a \prec a$ (reflexivity);*
- *$a \prec b$ and $b \prec a$ imply $a = b$ (antisymmetry);*
- *$a \prec b$ and $b \prec c$ imply $a \prec c$ (transitivity).*

A set S with a partial order is a chain iff $\forall a, b \in S$: $a \prec b$ or $b \prec a$.

DEFINITION 3.4. *A partial order pattern P is a set of event sets $\{p_i\}$,$i = 1, \ldots, n$ together with a partial order $\prec$ over them.*

DEFINITION 3.5. *A sequential pattern P is a partial order pattern that is a chain: $p_1 \prec p_2 \prec \ldots \prec p_n$.*

Note that in Episode mining sequential patterns are called serial patterns.

DEFINITION 3.6. *A parallel pattern P is a partial order pattern with no order relations among the event sets.*

DEFINITION 3.7. *A sequence $S = \{s_i\}, i = 1, ..., k$ matches a sequential pattern $P = \{p_j\}, i = 1, ..., m$ (or a pattern occurs in the sequence) iff $\exists i_1, ..., i_m$ with $p_j \subseteq s_{i_j}$ for $j = 1, ..., m$, such that $\forall 1 \leq j, k \leq m$: $p_j \prec p_k$ implies $i_j < i_k$. We will denote such a match by $m_{i_1,i_m}(P, S)$.*

DEFINITION 3.8. *A match $m_{i_1,i_m}(P, S)$ is the earliest match iff for any other match $m_{j_1,j_m}(P, S)$ $i_k \leq j_k$, $\forall k = 1, \ldots, m$,*

DEFINITION 3.9. *A pattern P has support$(P) = s$ in an SDB D if D contains s distinct event sequences that match P. A pattern is frequent iff its support is no less than some predefined minimum support value μ, i.e. support$(P) >= \mu$.*

In the following, when talking about patterns, we will always assume that they are frequent, with some minimum support μ defined.

DEFINITION 3.10. *A frequent pattern P is considered closed in an SDB D, if is there is no pattern $P' \neq P$ in D, such that $\exists m(P, P')$ (i.e. P occurs in P') and support(P') = support(P).*

DEFINITION 3.11. *A pattern P is considered margin-closed in an SDB D, with margin α, if is there is no pattern $P' \neq P$ in D such that P occurs in P' and support$(P') > (1-\alpha) *$ support(P).*

In other words, P is margin-closed if there is no pattern P' that contains P and is almost as frequent.

In order to describe the algorithms in the following sections, we need to introduce the notion of *projected databases*, which is extremely useful in constructing efficient algorithms for sequential pattern mining.

DEFINITION 3.12. *Given a pattern $P = \{p_i\}$, $i = 1, \ldots, |P|$ and a sequence $S = \{s_j\}$,$j = 1, \ldots, |S|$, with the earliest match $m_{k_1,k_m}(P, S)$, a projection of S on P results in a projected sequence $S|P = \{s_t\}$, where $t = k_m + 1, \ldots, |S|$. We refer to k_m as an offset.*

DEFINITION 3.13. *Given a pattern $P = \{p_i\}$, $i = 1, \ldots, |P|$ and an SDB $D = \{S_j\}$,$j = 1, \ldots, |D|$, a projection of D on P is a projected database $D|P$, consisting of projected sequences $S_{j_u}|P$, obtained by projecting S_{j_u} onto P. Note that if a sequence does not match a pattern, it does not appear in the projected database.*

Projected database $D|P$ can be efficiently represented with a list of pairs of indices (j_u, t_u), where j_u refers to $S_{j_u}|P$ and t_u is the corresponding offset.

DEFINITION 3.14. *For a projected sequence $S|P$ we define two operations:*

- $original(S|P) = S$*; and*
- $prefix(S|P) = \{s_t\}$, $t = 1, \ldots, k_m$, *where $m_{k_1,k_m}(P, S)$ is the earliest match.*

in other words, $original$ of a projection returns the whole sequence S, while $prefix$ of a projection returns the part of the sequence preceding the projection.

3.2 The BIDE algorithm

BIDE is an efficient algorithm for finding frequent closed sequential patterns in sequential databases [39]. We extend this algorithm to enforce the margin-closed constraints. In order to make this paper self-contained we provide a detailed description of BIDE using our own definitions.

BIDE is initially called with the full sequential database D, minimum support μ and an empty pattern $P = \emptyset$. It returns a list of frequent closed sequential patterns. BIDE operates by recursively extending patterns, and, while their frequency is above the minimum support, checking closure properties of the extensions.

Consider a frequent pattern $P = \{p_i\}$,$i = 1, \ldots, n$. There are two ways to extend pattern P *forward* with item j:

- Appending the set $p_{n+1} = \{j\}$ to P obtaining $P' = p_1 \ldots p_n p_{n+1}$, called a forward-S(equence)-extension.
- Adding j to the last itemset of P: $P' = p_1 \ldots p'_n$, with $p'_n = p_n \cup j$, assuming $j \notin p_n$, called a forward-I(tem)-extension.

Similarly, a pattern can be extended *backward*

- Inserting the set $p_x = \{j\}$ into P anywhere before the last set obtaining $P' = p_1 \ldots p_i p_x p_{i+1} \ldots p_n$, for some $0 \leq i \leq n$, called a backward-S(et)-extension.
- Adding j to any set in P obtaining $P' = p_1 \ldots p'_i \ldots p_n$, with $p'_i = p_i \cup j$, assuming $j \notin p_n$, $1 \leq i \leq n$, called a backward-I(tem)-extension.

According to a Theorem 3 of [40], a pattern is closed if there exists no forward-S-extension item, forward-I-extension item, backward-S-extension item, nor backward-I-extension item with the same support.

Furthermore, if there is a backwards extension item, then the resulting extension and all of its future extension are explored in a different branch of recursion, meaning that it can be pruned from current analysis. These insights are combined in BIDE, leading to a very memory-efficient algorithm, because the patterns found do not need to be kept in memory while the algorithm is running.

Algorithm 1 BIDE Algorithm

Require: Sequential Pattern $P = \{p_i\}$, Projected Database $D|P$, minimum support μ

```
1: F - set of frequent closed patterns
2: l = |P|
3: Ls = sStepFrequentItems(P, D|P, μ);
4: Li = iStepFrequentItems(P, D|P, μ);
5: if !(frequencyCheck(Ls, P) || frequencyCheck(Li, P)) then
6:    if backscan(P', D', true) then
7:       F = F ∪ P
8:    end if
9: end if
10: for itemset p ∈ Ls do
11:    P' = p_1, .., p_l, p
12:    if backscan(P', D|P', false) then
13:       F = F ∪ bide(P', D', μ);
14:    end if
15: end for
16: for itemset p ∈ Li do
17:    P' = p_1, .., p_{l-1}, p_l ∪ p
18:    if backscan(P', D|P', false) then
19:       F = F ∪ bide(P', D', μ);
20:    end if
21: end for
22: return F
```

Algorithm 2 FrequencyCheck

Require: Pattern P, HashMap M of forward (I or S) extension items with their supports

```
1: for i ∈ M do
2:    if M(i)=support(P) then
3:       return true
4:    end if
5: end for
6: return false
```

Specifically, consider pseudo-code for BIDE (Algorithm 1). In Lines 3-4 items that can be used in forward extension of the current pattern are found. If there is no forward extension with the same support (Line 5), the backward closure is checked (Line 6) using function backScan. If the pattern is also backwards-closed, it can be added to the set of closed frequent patterns (Line 7).

Then, we check every item in forward S and I extensions (in the two for-loops) to see whether it is explored in a different branch of recursion, again via backScan function (Lines 12 and 18). If not, then we project the database on the extension and call BIDE recursively on the extension and the new projected database.

Pseudo-code for $sStepFrequentItems$ and $iStepFrequentItems$ is shown in Algorithms 3 and 4. Algorithm 2 shows the FrequencyCheck function. These functions are rather straightforward.

It remains to discuss the $backScan$ function (Algorithm 5). The backScan function has two uses. The first time it is called in function BIDE, closure check flag is set to true (Line 6). Then backScan returns true if and only if pattern P is backwards closed, i.e. if there is no backwards extension with the same support. This is **Case I**. The other calls from BIDE are with closure check set to false. In these situations backScan needs to check if a pattern extension is backwards closed in its projected database i.e. that it can't be

Algorithm 3 Finding forward-S-expansion Candidates

Require: Sequential Pattern $P = \{p_i\}$; Projected Database $D|P$, minimum support μ

Initialize Hash Map M
for $i = 1, \ldots, |D|P|$ **do**
  $I = \emptyset$
  for $j = 1, \ldots, |s_i|$ **do**
    $I = I \cup s_{ij}$
  end for
  for item $\in I$ **do**
    $M(i) = M(i) + 1$
  end for
end for
for $i \in M$ **do**
  if $M(i) < \mu$ **then**
    Delete $M(i)$
  end if
end for
return M

Algorithm 4 Finding forward-I-expansion Candidates

Require: Sequential Pattern $P = \{p_i\}$; Projected Database $D|P$, μ

Initialize Hash Map M
Let $l = |P|$
for $i = 1, \ldots, |D|P|$ **do**
  $I = \emptyset$
  for $j = 1, \ldots, |s_i|$ **do**
    if $p_l \in s_i$ **then**
      $I = I \cup s_{ij}$
    end if
  end for
  for item $i \in I$ **do**
    $M(i) = M(i) + 1$
  end for
end for
for $i \in M$ **do**
  if $M(i) < \mu$ **then**
    Delete $M(i)$
  end if
end for
return M

reached in a different way, via a different recursion branch. This is **Case II**.

In order to check for backward extensions of a pattern P, we need to know which parts of sequences in D need to be looked at. If P has a backward-S-extension item t between p_i and $p_i + 1$, it means that in each sequence $S \in D$ matching P there is a particular match $m_{k_1,k_m}(P, S)$, such that t occurs between s_{k_i} and $s_{k_{i+1}}$. In order to check for an existence of such an item, we can find the earliest and the latest matches $m_{k_1,k_m}(P, S)$ and $m_{j_1,j_m}(P, S)$, and examine the itemsets $s_{k_i+1}, \ldots, s_{j_{i+1}-1}$. In other words, we check the itemsets between the earliest occurrence in a match of p_i and the latest occurrence in a match of p_{i+1}. Similarly, we can check for existence of a backward-I-extension item t by looking at all possible occurrences of t together with p_i, starting from earliest and ending with latest match occurrences of p_i. Function $FindMaximumGaps$ (Algorithm 6) is used exactly for finding and storing the earliest and latest indices of consecutive itemsets of pattern P in a match $m(P, S)$. Finding of the latest match is

Algorithm 5 BackScan Function. If $closedCheck$ is true, checks if P is closed. If $closedCheck$ is false, checks if P is examined in a different branch of the recursion.

Require: Sequential Pattern $P = \{p_i\}$, Projected Database $D|P$, μ,$closedCheck$

Initialize a 3D integer array G (for gaps)
for $i = 1, \ldots, |D|$ **do**
  if $closedCheck$ **then**
    $G[i] = FindMaximumGaps(P, original(S_i))$
  else
    $G[i] = FindMaximumGaps(P, prefix(S_i))$
  end if
end for
if $BackwardIExpansionCheck(P, D|P, \mu, closedCheck, G)$ **then**
  if $BackwardSExpansionCheck(P, D|P, \mu, closedCheck, G)$ **then**
    return true
  end if
end if
return false

Algorithm 6 FindMaximumGaps

Require: Sequential Pattern $P = \{p_i\}$; Event Sequence S

Initialize $|P| \times 2$ integer array G
if $\exists m_{i_1,i_m}(P, S)$ **then**
  $G[0][0] = 0$
  for $j = 1, \ldots, |P|$ **do**
    Set $G[j][0] = i_j$
  end for
  Compute S^r - reverse of S
  Compute P^r - reverse of P
  Compute $m_{i_1,i_m}(P^r, S^r)$
  for $j = 1, \ldots, |P^r|$ **do**
    Set $G[|P| - j][1] = |S| - i_j$
  end for
end if
return G

most efficiently found by searching for a reverse of P in a reverse of sequence S, and transforming the indices appropriately.

We can now discuss the two Cases mentioned above. In **Case I** closure check flag is set to true. Since we want to check if pattern P is closed, we need to fully examine all sequences in $D|P$ for potential extensions. Therefore, function FindMaximumGaps is called on original sequences in $D|P$, not on the projections. In **Case II**, we only care if there is a backward extension in order to prune the current pattern. Therefore, when $closedCheck$ is false, FindMaximumGaps is called on prefixes of sequences in $D|P$.

Once array G is computed, we check for I-expansions and for S-expansions (Algorithms 7 and 8). Consider BackwardIExpansionCheck. We want to detect if an item can be inserted into any itemset of P, while maintaining the same support. Therefore, for each position in P, we examine all sequences in $D|P$, in the intervals specified by array G. If $closedCheck$ is true, we look at the full sequence S, otherwise we look at the prefix of $S|P$. The 'end' indices need to be computed differently for the two cases, because when $closedCheck$ is false, the last occurrence of the last itemset of P is also the first potential point for forward expansion and does not need to be considered, but when $closedCheck$ is true,

we check for closedness of P and all potential expansion locations need to be examined.

For each s_j, if $p_i \in s_j$, we add all items in s_j to set C, except for items already in p_i. After processing a sequence, we update frequency counts of items in C that are stored a hash map M, and keep track of the maximum frequency value. Once we have processed all sequences for a particular p_i, we check if the maximum frequency is equal to support of P ($support(P) = |D|P|$). If so, that means that there is some backward-I-expansion item for P, and therefore P is not closed and, there is another recursion branch that will examine this expansion, so BackwardIExpansionCheck returns false. If maximum frequency is below support of P, BackwardIExpansionCheck return true.

BackwardSExpansionCheck operates similarly.

Algorithm 7 BackwardIExpansionCheck

Require: Pattern P, Projected Database $D|P$, μ,$closedCheck$, gap array G

```
1: for i = 1, ..., |P| do
2:    Initialize Hash Map M
3:    for j = 1, ..., |D| do
4:       start = G[j][i][0] + 1
5:       if closedCheck then
6:          end = G[j][i][1]
7:          S' = original(S_j)
8:       else
9:          end = G[j][i][1] − 1
10:         S' = prefix(S_j)
11:      end if
12:      C = ∅
13:      for k = start, ..., end do
14:         if P_i ∈ S'_k then
15:            C = C ∪ S'_k
16:         end if
17:      end for
18:      C = C − P_i
19:      max = 0
20:      for s_i ∈ C do
21:         M(s_i) = M(s_i) + 1
22:         if M(s_i) > max then
23:            max = M(s_i)
24:         end if
25:      end for
26:   end for
27:   if max = support(P) then
28:      return false;
29:   end if
30: end for
31: return true
```

3.3 The BIDE-Margin algorithm

We now describe the changes required to enforce margin-closedness in BIDE leading to the BIDE-Margin algorithm. The flag $marginCheck$ is used in the backScan function instead of $closedCheck$ and there is the additional margin parameter α. There are three changes to the functions described in the following sections.

When Forward Expansion is considered, rather than checking if there are items with the same support as the current pattern, one instead checks for presence of items that are within margin α of the pattern's support. The function FrequencyCheck for BIDE-Margin is shown in Algorithm 9.

Algorithm 8 BackwardSExpansionCheck

Require: Pattern P, Projected Database $D|P$, μ,$closedCheck$, gap array G

```
1: for i = 1, ..., |P| do
2:    Initialize Hash Map M
3:    for j = 1, ..., |D| do
4:       start = G[j][i][0] + 1
5:       end = G[j][i][1] − 1
6:       if closedCheck then
7:          S' = original(S_j)
8:       else
9:          S' = prefix(S_j)
10:      end if
11:      C = ∅
12:      for k = start, ..., end do
13:         C = C ∪ S'_k
14:      end for
15:      max = 0
16:      for s_i ∈ C do
17:         M(s_i) = M(s_i) + 1
18:         if M(s_i) > max then
19:            max = M(s_i)
20:         end if
21:      end for
22:   end for
23:   if max = support(P) then
24:      return false;
25:   end if
26: end for
27: return true
```

Algorithm 9 FrequencyCheck for BIDE-Margin

Require: Pattern P, HashMap M of forward (I or S) extension items with their supports, α

```
1: for i ∈ L do
2:    if M(i) >= (1 − α) * support(P) then
3:       return true
4:    end if
5: end for
6: return false
```

The other two changes involve checking backward closure. We need to check if there are any items that are margin-close to the pattern, and if so then the pattern is not margin-closed. This leads to changes to BackwardIExpansionCheck and BackwardSExpansionCheck functions. Algorithms 10 and 11 respectively show how these algorithms need to be modified for BIDE-Margin. The parameter $marginCheck$ replaces $closedCheck$, and is used similarly, except for additions in Lines 25-29 and Lines 21-25. When $marginCheck$ is true we check if pattern is margin-closed, and therefore if there is an item with frequency above μ and within margin of the support of P, we know that P is not margin-closed. Note that when $marginCheck$ is false, this check should not be performed - we cannot disregard the recursion branches going from the current pattern unless there is a backward extension with exactly the same support.

3.4 Computational Efficiency

The BIDE-Margin algorithm has the same complexity as BIDE, since it still generates all frequent sequential patterns, in exactly the same fashion. However, unlike BIDE it searches for margin-

Algorithm 10 BackwardIExpansionCheck for BIDE-Margin

Require: P, $D|P$, μ,$marginCheck$, G, α

for $i = 1, \ldots, |P|$ **do**
  Initialize Hash Map M
  for $j = 1, \ldots, |D|$ **do**
    $start = G[j][i][0] + 1$
    if $marginCheck$ **then**
      $end = G[j][i][1]$
      $S' = original(S_j)$
    else
      $end = G[j][i][1] - 1$
      $S' = prefix(S_j)$
    end if
    $C = \emptyset$
    for $k = start, \ldots, end$ **do**
      if $P_i \in S'_k$ **then**
        $C = C \cup S'_k$
      end if
    end for
    $C = C - P_i$
    $max = 0$
    for $s_i \in C$ **do**
      $M(s_i) = M(s_i) + 1$
      if $M(s_i) > max$ **then**
        $max = M(s_i)$
      end if
      if $marginCheck$ **then**
        if $max \geq (1 - \alpha) * |D|P|$ AND $max \geq \mu$ **then**
          return false
        end if
      end if
    end for
  end for
  if $max = |D|$ **then**
    return false;
  end if
end for
return true

Algorithm 11 BackwardSExpansionCheck for BIDE-Margin

Require: P, $D|P$, μ,$marginCheck$, G, α

for $i = 1, \ldots, |P|$ **do**
  Initialize Hash Map M
  for $j = 1, \ldots, |D|$ **do**
    $start = G[j][i][0] + 1$
    $end = G[j][i][1] - 1$
    if $marginCheck$ **then**
      $S' = original(S_j)$
    else
      $S' = prefix(S_j)$
    end if
    $C = \emptyset$
    for $k = start, \ldots, end$ **do**
      $C = C \cup S'_k$
    end for
    $max = 0$
    for $s_i \in C$ **do**
      $M(s_i) = M(s_i) + 1$
      if $M(s_i) > max$ **then**
        $max = M(s_i)$
      end if
      if $marginCheck$ **then**
        if $max \geq (1 - \alpha) * |D|P|$ AND $max \geq \mu$ **then**
          return false
        end if
      end if
    end for
  end for
  if $max = |D|$ **then**
    return false;
  end if
end for
return true

closed, rather than just closed, patterns and therefore it will need to check closeness less frequently. In other words, due to a "looser" frequency check used by BIDE-Margin (Algorithm 9), the call to $backscan$ algorithm in Line 6 of Algorithm 1 will occur less frequently in BIDE-Margin than in BIDE. Thus, while the overall algorithm complexity is the same, BIDE-Margin may perform slightly faster. The extent of this depends on the nature of the data and the value of margin specified.

We would also like to note that BIDE-Margin is significantly more efficient than a brute force post-processing of BIDE results would be. If N is the number of patterns produced by BIDE for a particular value of support, the postprocessing would require $O(N)$ memory and $O(N^2)$ time to order the patterns by their support and then to check for each pattern P if it is margin-closed.

4. EXPERIMENTS

We performed experiments on real life sequential data sets to compare BIDE and BIDE-Margin in two ways: 1) The number of patterns produced. By definition BIDE-Margin with $\alpha > 0$ produces the same or less patterns than BIDE. The goal of the experiment is to quantify the extent of this reduction on real life data. 2) Predictiveness of patterns found: We compare the classification performance of the sets of patterns when used as features in SVM training. Since BIDE-Margin suppresses only features that are very similar in frequency to reported features, we expect to see only minor performance decrease, if any. With SVM being a classifier that can deal with high dimensional data and redundancy among the features this is a tough test.

We did not perform run-time comparisons between BIDE and BIDE-Margin, since the differences are expected to be small. Similarly, the scalability with the number of sequences in the database is inherited directly from BIDE.

4.1 Data

We performed experiments on six interval datasets, previously used in [29], summarized in Table 1. While technically databases of intervals, they can be interpreted as sequential databases by treating start and end boundaries of an interval as separate events [42]. Specifically, each symbolic interval, a triple (t_s, t_e, σ) with event $\sigma \in \Sigma$ and time stamps $t_s \leq t_e$, is converted into two symbolic time points (t_s, σ^+) and (t_e, σ^-), and then all time points with the same time stamp are aggregated into itemsets, resulting in a standard event sequence as in Definition 3.1. Further details are given in [29].

The advantage of this collection is that class labels are available for each sequence that allows an automated evaluation of patterns using a classifier, while the categorical sequential data available in the UCI Machine Learning Repository [3] is largely unlabeled such as web log data.

Data	Intervals	Labels	Sequences	Classes
ASL-BU	18250	154	441	7
Auslan2	900	12	200	10
Blocks	1207	8	210	8
Context	12916	54	240	5
Pioneer	4883	92	160	3
Skating	18953	41	530	6/7

Table 1: Interval data: Seven databases consisting of many sequences of labeled intervals with class labels for each sequence.

ASL-BU[1] The intervals are transcriptions from videos of American Sign Language expressions provided by Boston University [30]. It consists of observation interval sequences with labels such as *head mvmt: nod rapid* or *shoulders forward* that belong to one of 7 classes like *yes-no question* or *rhetorical question*.

Auslan2 The intervals were derived from the high quality Australian Sign Language dataset in the UCI repository [3] donated by Kadous [19]. The x,y,z dimensions were discretized using Persist with 2 bins, 5 dimensions representing the fingers were discretized into 2 bins using the median as the divider. Each sequence represents a word like *girl* or *right*.

Blocks[2] The intervals describe visual primitives obtained from videos of a human hand stacking colored blocks provided by [15]. The interval labels describe which blocks touch and the actions of the hand (*contacts blue red*, *attached hand red*). Each sequence represents one of 8 different scenarios from atomic actions (*pick-up*) to complete scenarios (*assemble*).

Context[3] The intervals were derived from categoric and numeric data describing the context of a mobile device carried by humans in different situations [23]. Numeric sensors were discretized using 2-3 bins chosen manually based on exploratory data analysis. Each sequence represents one of five scenarios such as *street* or *meeting*.

Pioneer The intervals were derived from the Pioneer-1 datasets in the UCI repository [3]. The numerical time series were discretized into 2-4 bins by choosing thresholds manually based on exploratory data analysis. Each sequence describes one of three scenarios: *gripper*, *move*, *turn*.

Skating The intervals were derived from 14 dimensional numerical time series describing muscle activity and leg position of 6 professional In-Line Speed Skaters during controlled tests at 7 different speeds on a treadmill [27]. The time series were discretized into 2-3 bins using Persist and manually chosen thresholds. Each sequence represents a complete movement cycle and is labeled by skater or speed.

4.2 Numerosity

By construction, the number of patterns produced by BIDE-Margin is always less than or equal to that produced by BIDE. Figure 1 shows the number of patterns (on a log10 scale) found by these methods using different support thresholds and margin values. For all datasets except ASL-BU, BIDE-Margin produces significantly fewer patterns. The reduction is strongest for the Context and Skating data and for Auslan2 for large minimum supports.

4.3 Predictiveness

Patterns obtained by unsupervised mining can be used for knowledge discovery by ranking and analyzing them directly, for generation of temporal association rules [18], or as features in predictive models [11]. We analyzed the predictiveness of the patterns by estimating classifier performance with each set of patterns.

In our experiments we have used the Spider Toolbox for Matlab[4] As classifier, we focused on Support Vector Machines. We have also experimented with decision trees and random forests, obtaining qualitatively similar results.

Once patterns were generated, for a particular value of support and margin, we have performed 10-fold cross-validation with linear SVM, setting parameter C in turn to 2^k, $k = -10, -9, \ldots, 9, 10$. The best value over all values of C is reported. Note, that this is done purely for the purpose of comparing the properties of two unsupervised pattern mining techniques, hence we did not interleave the pattern mining with the cross validation, as would be needed if the goal were to train a classifier with good generalization performance.

The results are shown in Figure 2. The y-axis is the lowest classification error, while the x-axis is the minimum support. The results with different margin values are shown as different lines. Examination of these results suggests that using margin 0.05 or 0.1 barely affects the classification error rate. Margin of 0.2 does lead to noticeably worse results on Pioneer dataset, and on Auslan2 with support 20, but not on the other datasets. The differences in performance tend to become smaller as support increases and the number of patterns decreases.

Figure 3 shows results obtained with J48, using the default settings. The results are qualitatively similar to those obtained with SVM, i.e. classification error does not increase for small values of the margin. Results with random forests are omitted due to space constraints.

5. CONCLUSION

We presented a new constraint for reducing the output of sequential pattern mining and an efficient algorithm for mining such patterns. We have demonstrated that the number of margin-closed patterns can be a lot smaller than that of closed patterns, but that these patterns are just as useful, as evidenced by performance of classifiers built using these patterns.

Using data mining in real life systems often requires the analyst to understand and trust the reported results to take appropriate action. We believe that reporting of exact patterns with exact frequency and favoring longer patterns while pruning similar shorter patterns are all advantageous for interpretation of mining results by an analyst. For some domains, however, error tolerance [45] may be more important than interpretability.

Mining closed sequential patterns is an important task in temporal data mining from time point and time interval data [28] and it is a substep in the process of mining partial orders [9]. In future work, we intend to conduct an experimental evaluation of the run-time of BIDE-Margin compared with BIDE using both actual pattern mining time and the time needed by follow up data mining tasks such as classification or grouping of sequences into partial orders.

6. REFERENCES

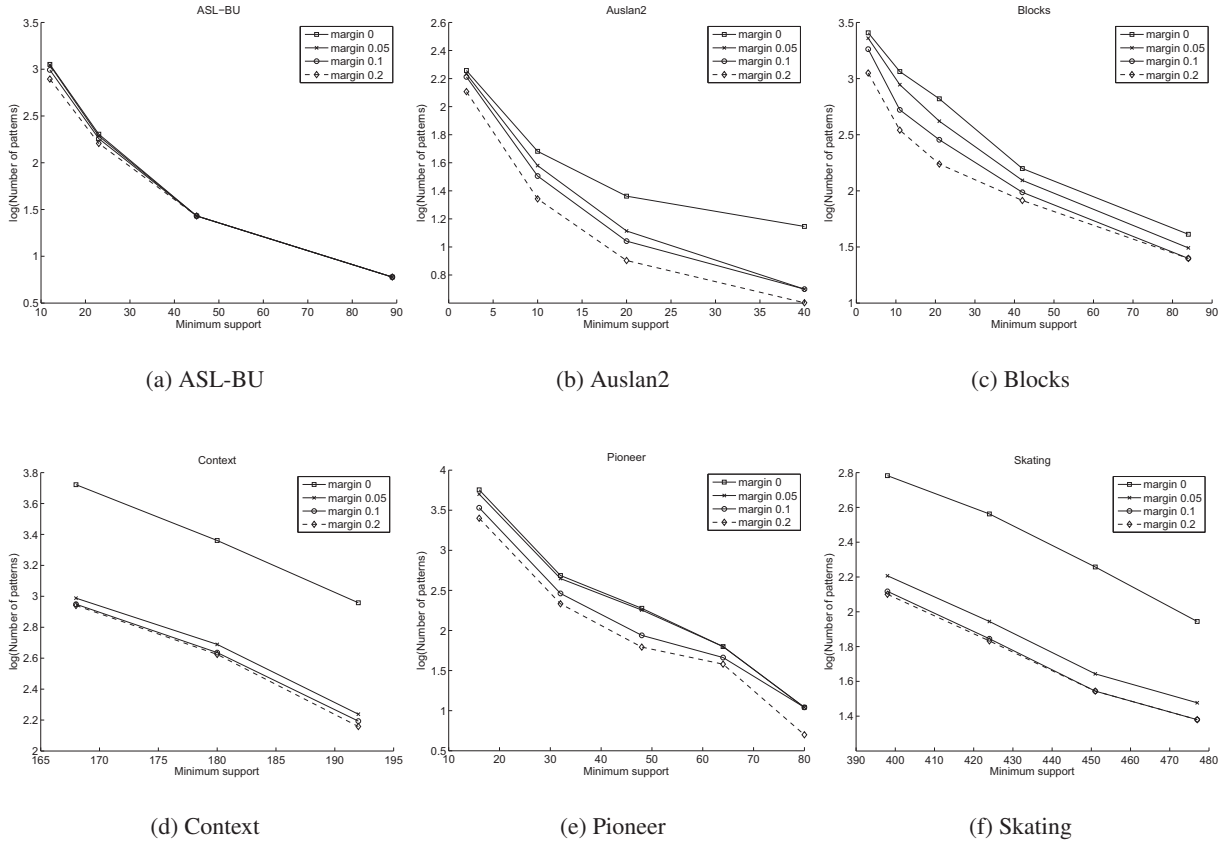
[1] F. Afrati, A. Gionis, and H. Mannila. Approximating a collection of frequent sets. In *Proc. 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 12–19. ACM Press, 2004.

[1] http://www.bu.edu/asllrp/

[2] ftp://ftp.ecn.purdue.edu/qobi/ama.tar.Z

[3] http://www.cis.hut.fi/jhimberg/contextdata/index.shtml

[4] http://www.kyb.mpg.de/bs/people/spider/main.html



(a) ASL-BU (b) Auslan2 (c) Blocks

(d) Context (e) Pioneer (f) Skating

Figure 1: Comparison of the number of patterns (log10 scale) for different minimum support thresholds and margin values.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proc. IEEE ICDE*, pages 3–14. IEEE Press, 1995.

[3] A. Asuncion and D.J. Newman. UCI Machine Learning Repository. University of California, Irvine `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[4] F. Bonchi and C. Lucchese. On condensed representations of constrained frequent patterns. *Knowl. Inf. Syst.*, 9(2):180–201, 2006.

[5] J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pages 62–73, 2000.

[6] Björn Bringmann and Albrecht Zimmermann. One in a million: picking the right patterns. *Knowledge and Information Systems*, 18(1):61–81, 2008.

[7] T. Calders and B. Goethals. Non-derivable itemset mining. *Data Mining and Knowledge Discovery*, 14(1):171–206, 2007.

[8] T. Calders, C. Rigotti, and J.-F. Boulicaut. A survey on condensed representations for frequent sets. In *Constraint-Based Mining and Inductive Databases*, pages 64–80, 2006.

[9] G. Casas-Garriga. Summarizing sequential data with closed partial orders. In *Proc. of the 5th SIAM Intl. Conf. on Data Mining (SDM)*, pages 380–391. SIAM, 2005.

[10] Lei Chang, Tengjiao Wang, Dongqing Yang, Hua Luan, and Shiwei Tang. Efficient algorithms for incremental maintenance of closed sequential patterns in large databases. *Data Knowl. Eng.*, 68(1):68–106, 2009.

[11] H. Cheng, X. Yan, J. Han, and C.-W. Hsu. Discriminative frequent pattern analysis for effective classification. In *Proc. IEEE ICDE*, 2007.

[12] J. Cheng, Y. Ke, and W. Ng. δ-tolerance closed frequent itemsets. In *Proc. 6th IEEE Int. Conf. on Data Mining*, pages 139–148. IEEE Press, 2006.

[13] L. De Raedt, T. Guns, and S. Nijssen. Constraint programming for itemset mining. In *Proc. 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 204–212. ACM, 2008.

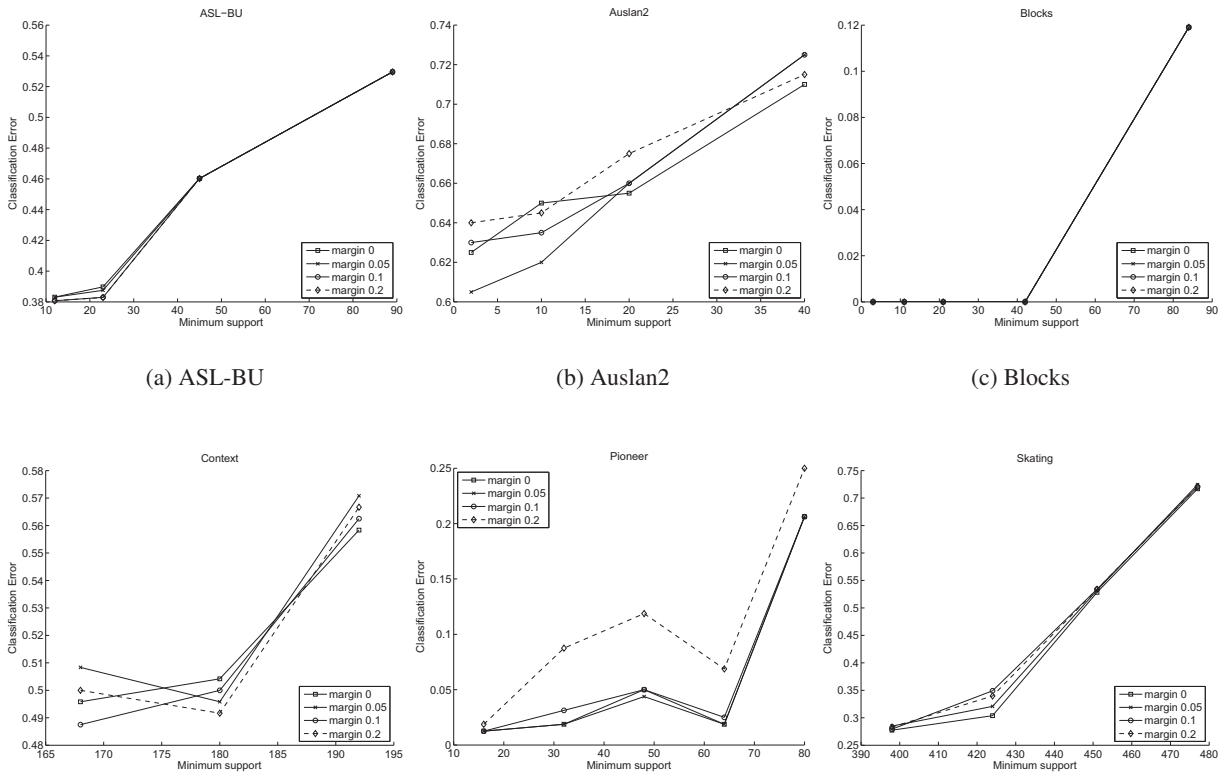
[14] G. Dong and J. Pei. *Sequence Data Mining*. Morgan Kaufmann, 2007.

[15] A. Fern. *Learning Models and Formulas of a Temporal Event Logic*. PhD thesis, Purdue University, West Lafayette, IN, USA, 2004.

[16] R. Gupta, G. Fang, B. Field, M. Steinbach, and V. Kumar. Quantitative evaluation of approximate frequent pattern mining algorithms. In *Proc. 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 301–309. ACM, 2008.

[17] J. Han and M. Kamber. *Data Mining - Concepts and Techniques, 2nd edition*. Morgan Kaufmann, 2006.

[18] Frank Höppner. Discovery of temporal patterns - learning rules about the qualitative behaviour of time series. In *Proc.*



(a) ASL-BU (b) Auslan2 (c) Blocks

(d) Context (e) Pioneer (f) Skating

Figure 2: SVM classification errors achieved with different minimum support thresholds and margin values.

of the 5th European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD), pages 192–203. Springer, 2001.

[19] M. W. Kadous. *Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series*. PhD thesis, University of New South Wales, 2002.

[20] Arno J. Knobbe and Eric K. Y. Ho. Pattern teams. In *PKDD*, pages 577–584, 2006.

[21] C. Lucchese, S. Orlando, and R. Perego. Fast and memory efficient mining of frequent closed itemsets. *IEEE TKDE*, 18(1):21–36, 2006.

[22] H. Mannila, H. Toivonen, and I. Verkamo. Discovery of frequent episodes in event sequences. In *Proc. of the 1st Intl. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 210–215. AAAI Press, 1995.

[23] J. Mäntyjärvi, J. Himberg, P. Kangas, U. Tuomela, and P. Huuskonen. Sensor signal data set for exploring context recognition of mobile devices. In *Proc. of 2nd Int. Conf. on Pervasive Computing (PERVASIVE 2004)*, pages 18–23. Springer, 2004.

[24] Fabian Moerchen, Michael Thies, and Alfred Ultsch. Efficient mining of all margin-closed itemsets with applications in temporal knowledge discovery and classification by compression. *Under review in Knowledge and Information Systems*.

[25] F. Mörchen. Algorithms for time series knowledge mining. In *Proc. 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 668–673. ACM Press, 2006.

[26] F. Mörchen. *Time Series Knowledge Mining*. PhD thesis, Philipps-University Marburg, Germany, 2006.

[27] F. Mörchen and A. Ultsch. Efficient mining of understandable patterns from multivariate interval time series. *Data Min. Knowl. Discov.*, 2007.

[28] Fabian Mörchen. Unsupervised pattern mining from symbolic temporal data. *SIGKDD Explor. Newsl.*, 9(1):41–55, 2007.

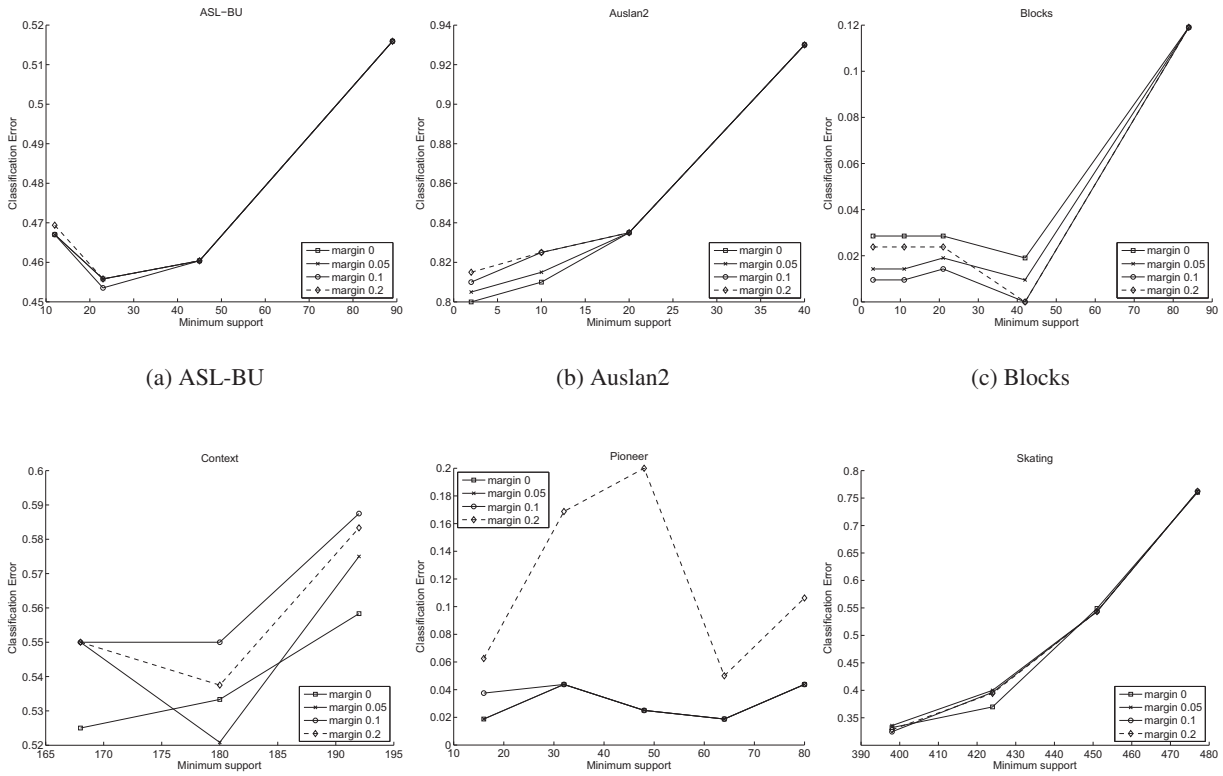
[29] Fabian Mörchen and Dmitriy Fradkin. Robust mining of time intervals with semi-interval partial order patterns. In *Proceedings of SIAM Conference on Data Mining (SDM)*, Columbus, Ohio, USA, 2010.

[30] P. Papaterou, G. Kollios, S. Sclaroff, and D. Gunopoulos. Discovering frequent arrangements of temporal intervals. In *Proc. of the 5th IEEE Intl. Conf. on Data Mining (ICDM)*, pages 354–361, 2005.

[31] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *Proceeding of the 7th Intl. Conf. on Database Theory (ICDT)*, pages 398–416. Springer, 1999.

[32] J. Pei, G. Dong, W. Zou, and J. Han. Mining condensed frequent pattern bases. *Knowledge and Information Systems*, 6(5):570–594, 2004.

[33] J. Pei, J. Han, and L. V. S. Lakshmanan. Mining frequent



(a) ASL-BU (b) Auslan2 (c) Blocks

(d) Context (e) Pioneer (f) Skating

Figure 3: J48 classification errors achieved with different minimum support thresholds and margin values.

itemsets with convertible constraints. In *Proc. IEEE Intl. Conf. on Data Engineering*. IEEE, 2001.

[34] Jian Pei, Haixun Wang, Jian Liu, Ke Wang, Jianyong Wang, and Philip S. Yu. Discovering frequent closed partial orders from strings. *IEEE TKDE*, 18(11):1467–1481, 2006.

[35] Marc Plantevit and Bruno Crémilleux. Condensed representation of sequential patterns according to frequency-based measures. In *IDA '09: Proceedings of the 8th International Symposium on Intelligent Data Analysis*, pages 155–166, Berlin, Heidelberg, 2009. Springer-Verlag.

[36] C. Raïssi, T. Calders, and P. Poncelet. Mining conjunctive sequential patterns. *Data Min. Knowl. Discov.*, 17(1):77–93, 2008.

[37] N. Tatti and J. Vreeken. Finding good itemsets by packing data. In *Proc. 8th IEEE Int. Conf. on Data Mining*, 2008.

[38] M. van Leeuwen, J. Vreeken, and A. Siebes. Compression picks item sets that matter. In *Proc. European Conf. on Principles and Practice of Knowledge Discovery in Databases*, pages 585–592, 2006.

[39] J. Wang and J. Han. BIDE: Efficient mining of frequent closed sequences. In *Proc. ICDE*, pages 79–90. IEEE Press, 2004.

[40] Jianyong Wang, Jiawei Han, and Chun Li. Frequent closed sequence mining without candidate maintenance. *IEEE Transactions on Knowledge and Data Engineering*, 19(8):1042–1056, 2007.

[41] Tao Wang. Compressing the set of frequent sequential patterns. In *FSKD '08: Proceedings of the 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, pages 330–334, Washington, DC, USA, 2008. IEEE Computer Society.

[42] Shin-Yu Wu and Yen-Liang Chen. Mining nonambiguous temporal patterns for interval-based events. *IEEE TKDE*, 19(6):742–758, 2007.

[43] M.J. Zaki. Mining non-redundant association rules. *Data Mining and Knowledge Discovery*, 9(3):223–248, 2004.

[44] Q. Zhao and S.S. Bhowmick. Sequential pattern mining: A survey. Technical report, Nanyang Technichal University, Singapore, 2003.

[45] Feida Zhu, Xifeng Yan, Jiawei Han, and Philip S. Yu. Efficient discovery of frequent approximate sequential patterns. In *ICDM '07: Proceedings of the Seventh IEEE International Conference on Data Mining*, pages 751–756, Washington, DC, USA, 2007. IEEE Computer Society.

Block Interaction: A Generative Summarization Scheme for Frequent Patterns

Ruoming Jin[1] Yang Xiang[2] Hui Hong[1] Kun Huang[2]
[1]Department of Computer Science, Kent State University, Kent, OH 44242
{jin, hhong}@cs.kent.edu
[2]Department of Biomedical Informatics, OSUCCC Biomedical Informatics Shared Resource,
The Ohio State University, Columbus, OH 43210
{yxiang,khuang@bmi.osu.edu}

ABSTRACT

Frequent pattern mining is an essential tool in the data miner's toolbox, with data applications running the gamut from itemsets, sequences, trees, to graphs and topological structures. Despite its importance, a major issue has clouded the frequent pattern mining methodology: the number of frequent patterns can easily become too large to be analyzed and used. Though many efforts have tried to tackle this issue, it remains to be an open problem. In this paper, we propose a novel *block-interaction* model to answer this call. This model can help summarize a collection of frequent itemsets and provide accurate support information using only a small number of frequent itemsets. At the heart of our approach is a set of core blocks, each of which is the Cartesian product of a frequent itemset and its support transactions. Those core blocks interact with each other through two basic operators (horizontal union and vertical union) to form the complexity of frequent patterns. Each frequent itemset can be expressed and its frequency can be accurately recovered through the combination of these core blocks. This is also the first complete generative model for describing the formation of frequent patterns. Specifically, we relate the problem of finding a minimal block-interaction model to a generalized set-cover problem, referred to as the graph set cover (GSC) problem. We develop an efficient algorithm based on GSC to discover the core blocks. A detailed experimental evaluation demonstrates the effectiveness of our approach.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Algorithms, Theory

UP'10, July 25th, 2010, Washington, DC, USA.

Keywords

block interaction, set cover with pairs, generative model, frequent itemsets, pattern summarization

1. INTRODUCTION

Frequent pattern mining is an essential tool in the data miner's toolbox, with data applications running the gamut from itemsets, sequences, trees, to graphs and topological structures [2, 26, 3, 32, 4]. Frequent pattern mining not only works by itself to discover hidden patterns from relational or structured data, but also serves as a basis for other data mining techniques, including association rules, clustering, classification, indexing, etc. Over the years, the data mining community has achieved great success in developing efficient and scalable mining algorithms to discover frequent patterns (See [13] for a state-of-the-art review).

However, a major issue has clouded the frequent pattern mining methodology from the very beginning: the number of frequent patterns can easily become too large to be analyzed and used. Many efforts have been made in attempting to reduce the number of frequent patterns, especially for itemsets (which can be generalized to other pattern types). An early reduction approach considered how to utilize "constraints" to filter out non-essential itemsets. Well-known examples include maximal frequent patterns [19], closed frequent patterns [21], non-derivable itemsets [9], and δ-cluster notation [29]. A weakness of this approach is that they either cannot recover the frequency of individual frequent itemsets (like maximal frequent patterns), or can still produce a very large number of patterns. Another approach, which includes top-k frequent patterns [14], top-k redundancy-aware patterns [28], and error-tolerant patterns [31], tries to rank the importance of individual patterns, or to revise the frequency concept to reduce the number of frequent patterns. However, these methods generally do not provide a good representation of the collection of frequent patterns.

The most recent approach is referred to as pattern summarization, which aims to offer a global view of the whole collection of frequent itemsets. In [1], the authors propose to use K itemsets as a concise representation to approximately cover the majority of the frequent itemsets. Typically, those itemsets are either maximal frequent itemsets or close to being maximal, with the condition that most of their subsets are frequent (subject to false positive constraint). A key open question mentioned in this work is how to derive the support information for an frequent itemset from such summarization. Several works have applied probabilistic inference to restore or browse the frequency of frequent itemsets [16, 30, 25,

22]. However, contrary to the original goal of [1], these works do not directly contribute to the reduction of frequent patterns.

Can we summarize a collection of frequent itemsets and provide accurate support information using only a small number of frequent itemsets? In this paper, we provide a novel *block-interaction* model to answer this call. At the heart of our approach is a set of core blocks, also referred to as hyperrectangles [27], or tiles [11, 12], each of which is the Cartesian product of a frequent itemset and its support transactions. Those core blocks interact with each other through two basic operators (horizontal union and vertical union) to form the complexity of frequent patterns. Each frequent itemset can be expressed and its frequency can be accurately recovered through the combination of these core blocks. This is the first complete generative model for describing the formation of frequent patterns. An additional benefit is that this model also provides a simple explanation of each frequent itemset based on a small set of core blocks and two basic operators. Finally, this generative model naturally reduces the entire collection of frequent itemsets to a very small core set. Note that in [18], we developed an approach to provide a generative view of an entire collection of itemsets. However, that model, similar to [1], could not recover support information for an individual itemset. Besides, the optimization goal of [18] is not to reduce the number of frequent itemsets to a small core set, but to minimize the representation or storage cost of maximal itemsets.

The contributions of this paper are as follows.

1. We develop the first generative model to describe the formation of frequent itemsets. This model not only reduces the entire collection of frequent itemsets to a small number of core patterns, but also it can express each frequent itemset and recover its frequency.
2. We relate the problem of finding a small set of core patterns to a generalized set-cover problem, referred to as the Graph Set Cover (GSC) problem. We develop an efficient algorithm utilizing GSC to compute the small set of core patterns.
3. We have performed a detailed experimental evaluation; our experimental results demonstrate the effectiveness of our approach.

2. BLOCK-INTERACTION MODEL AND PROBLEM DEFINITION

In this section, we formally introduce the *block-interaction* model for a collection of frequent itemsets F_α with α being the minimum support level.

Let the transaction database DB be represented as a binary matrix, i.e., a cell (i, j) is 1 if a transaction i contains item j; otherwise 0. For convenience, we also denote the database DB as the set of all cells which are 1, i.e., $DB = \{(i, j) : DB[i, j] = 1\}$. Let the block B be the Cartesian product of a transaction set T and an itemset I, i.e., $B = T \times I = \{(i, j) : i \in T \text{ and } j \in I\}$, such that $B \subseteq DB$. In other words, each block is an all-one submatrix of the binary matrix of DB. For a block B, we also denote $T(B)$ as the transaction set of B and $I(B)$ as the itemset of B, i.e., $T(B) = T$ and $I(B) = I$ for $B = T \times I$.

The block-interaction model contains a set of *core blocks* and two simple *block operators*, the block vertical union ($\oplus$) and the block horizontal union ($\ominus$) operators. Those core blocks and operators can provide a generative view of a collection of frequent itemsets and derive a concise explanation for each itemset.

Core Blocks: Given a transaction database DB, the block-interaction model contains a small set of core blocks, denoted as $\mathcal{B} = \{B_1, B_2, \cdots, B_p\}$, where B_i is a block of transaction database DB.

Block Vertical Union ($\oplus$): Given two blocks $B_1 = T_1 \times I_1$ and $B_2 = T_2 \times I_2$, the block vertical union operator generates a new block with the itemset being the *intersection* of two itemsets $I_1 \cap I_2$ and the transaction set being the *union* of two transaction sets $T_1 \cup T_2$ (Figure 1(a)):

$$B_1 \oplus B_2 = T_1 \times I_1 \oplus T_2 \times I_2 = (T_1 \cup T_2) \times (I_1 \cap I_2)$$

Block Horizontal Union ($\ominus$): Given two blocks $B_1 = T_1 \times I_1$ and $B_2 = T_2 \times I_2$, the block horizontal union operator generates a new block with the itemset being the *union* of two itemsets $I_1 \cup I_2$ and the transaction set being the *intersection* of two transaction sets $T_1 \cap T_2$ (Figure 1(b)):

$$B_1 \ominus B_2 = T_1 \times I_1 \ominus T_2 \times I_2 = (T_1 \cap T_2) \times (I_1 \cup I_2)$$

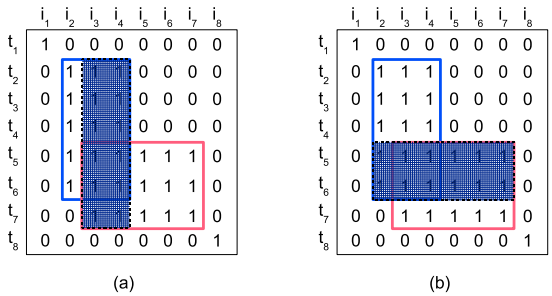


Figure 1: Block Union

Clearly, for any two blocks B_1 and B_2 of transaction database DB, their vertical union ($B_1 \oplus B_2$) and horizontal union ($B_1 \ominus B_2$) are also blocks of DB. It is also easy to observe the following properties of these two operators (the proof is omitted for simplicity).

LEMMA 1. *Both operators satisfy the commutative, associate, and distributive properties, i.e.,*

$$B_1 \oplus B_2 = B_2 \oplus B_1, \quad B_1 \oplus (B_2 \oplus B_3) = (B_1 \oplus B_2) \oplus B_3,$$
$$B_1 \ominus B_2 = B_2 \ominus B_1, \quad B_1 \ominus (B_2 \ominus B_3) = (B_1 \ominus B_2) \ominus B_3,$$
$$B_1 \oplus (B_2 \ominus B_3) = (B_1 \oplus B_2) \ominus (B_1 \oplus B_3),$$
$$B_1 \ominus (B_2 \oplus B_3) = (B_1 \ominus B_2) \oplus (B_1 \ominus B_3)$$

Given the core blocks $\mathcal{B}$ and the two block operators, we can recursively generate a large set of blocks. Formally, we introduce the closure of $\mathcal{B}$, denoted $\mathcal{P}(\mathcal{B})$, as the set of all blocks generated by the combination of core blocks $\mathcal{B}$ using the two operators:

Block Closure of $\mathcal{B}$, $\mathcal{P}(\mathcal{B})$: 1) for any block $B \in \mathcal{B}$, $B \in \mathcal{P}(\mathcal{B})$; 2) If $B_1 \in \mathcal{P}(\mathcal{B})$ and $B_2 \in \mathcal{P}(\mathcal{B})$, then $B_1 \oplus B_2 \in \mathcal{P}(\mathcal{B})$; 3) If $B_1 \in \mathcal{P}(\mathcal{B})$ and $B_2 \in \mathcal{P}(\mathcal{B})$, then $B_1 \ominus B_2 \in \mathcal{P}(\mathcal{B})$).

Now, we relate the frequent itemset $I \in F_\alpha$ to the set of core blocks $\mathcal{B}$ and its closure $\mathcal{P}$. We define $supp(I)$ to be the support of itemset I in the transaction database DB.

DEFINITION 1. *(**Block Support**) Given an itemset I, if a block B in $\mathcal{P}(\mathcal{B})$ subsumes I, i.e.,$I \subseteq I(B)$, and if $|T(B)| \geq (1 - \epsilon)supp(I)$, where ϵ is a user-preferred accuracy level for support recovery, then we say I is* **supported** *or* **explained** *by block B, denoted as $B \models I$. For a given set of frequent itemset F_α, and a set of core blocks $\mathcal{B}$, if any itemset I in F_α is supported by at least one block B in the closure $\mathcal{P}$ of $\mathcal{B}$, then we say that F_α is supported or explained by $\mathcal{B}$ or $\mathcal{P}$, denoted as $\mathcal{B} \models F_\alpha$ or $\mathcal{P} \models F_\alpha$.*

Given this, we can see that the block-interaction model provides a generative view of any frequent itemset by utilizing the two operators on top of a small set of core blocks. Indeed, each block in

the block closure can be written as an expression of core blocks and block operators. However, the potential problem is that such an expression may be too complex or too unnatural to provide explanatory power. For instance, a straightforward but extreme interaction model would be to consider each single item-transaction pair in the database as a core block, i.e., $\mathcal{B} = DB$. Alternatively, we could treat each single column (each item with all its support transactions) as a core block. Clearly, these sets of core blocks neither provide a satisfactory generative view of the collection of frequent itemsets nor are able to concisely summarize them. The underlying reason for such an issue is that the core blocks do not immediately relate to the frequent itemsets and the block expression can be too complex.

In order to derive a meaningful block-interaction model, we consider constraining the complexity of each block expression for supporting or explaining an itemset. Specifically, we introduce the concepts of (2×2)-*block support* (read "two by two") and (2×2)-*block interaction model*.

DEFINITION 2. *(*(2×2)**-Block Support***) Given an itemset I, if a block B in the block closure $\mathcal{P}$ supports I, i,e., $B \models I$, and if this block can be expressed in the following format,*

$$B = (B_1 \ominus B_2) \oplus (B_3 \ominus B_4), \tag{1}$$

where $B_1, B_2, B_3, B_4 \in \mathcal{B}$ are core blocks, then we say I is (2×2)-block supported by $\mathcal{B}$. If each itemset I in F_α is (2×2)-block supported by $\mathcal{B}$, then we say $\mathcal{B}$ is a (2×2)-block interaction model for F_α.

Note that in our (2×2)-block support definition, the four blocks, B_1, B_2, B_3 and B_4, are not necessarily different. In other words, some or all may correspond to the same block. For instance, B_1 and B_2 can be the same: $B_1 = B_2$. Figure 2 illustrates the block expression for the 2×2 block support (and interaction model). Note that we could also define the 2×2 block support based on the block expression $(B_1 \oplus B_2) \ominus (B_3 \oplus B_4)$. The methodology developed in this paper can be naturally adopted for such a model as well. Due to space limitation, we focus our discussion the first model. In general, we may further relax the constraint to consider more operators in the block expression. However, relaxation will increase the expression complexity. The 2×2 model is the most concise one which allows both operators to be utilized in the block expression in a symmetric fashion.

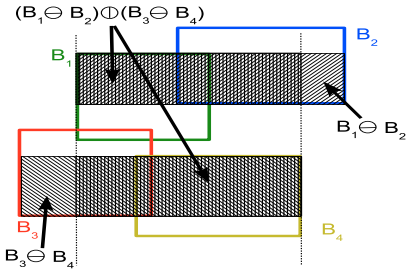


Figure 2: 2×2 **block interaction** $(B_1 \ominus B_2) \oplus (B_3 \ominus B_4)$

Given the (2×2) block interaction model, our goal is to provide a generative view of an entire collection of itemsets F_α using only a small set of core blocks $\mathcal{B}$.

DEFINITION 3. **(Minimal** (2×2)**-Block Interaction Model)** *Let the complexity of the (2×2)-block interaction model be defined as the number of core blocks, $|\mathcal{B}|$. Given a collection of frequent itemset F_α, we seek the most concise (2×2)-block interaction model to explain F_α, i.e.,*

$$\arg\min_{|\mathcal{B}|} \mathcal{B} \models F_\alpha$$

3. THEORETICAL BASIS

In this section, we first study the NP-hardness of the *minimal* (2×2)*-block interaction model* (Subsection 3.1) and then we introduce a new variant of the set cover problem, referred to as the *graph set cover* problem (GSC for short), which forms the basis of our algorithm to identify the minimal (2×2)-block interaction model (Subsection 3.2). In the following sections (Section 4 and Section 5, we discuss how to transform our problem into GSC problem and how to solve it efficiently.

3.1 Hardness Results

THEOREM 1. *Given a transaction database DB and a collection of frequent itemsets F_α, it is NP-hard to find a minimal (2×2)-block interaction model.*

Proof sketch of Theorem 1 can be found in our technical report [17].

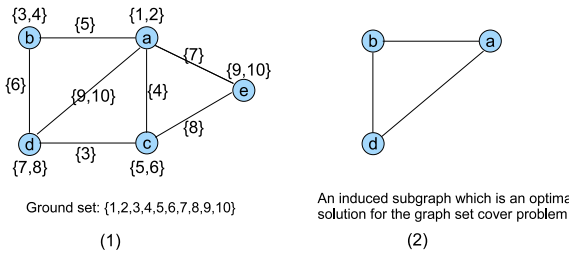


Figure 3: An example of GSC problem

3.2 Graph Set Cover Problem

In this subsection, we introduce a new variant of the set cover problem, which is closely related to the minimal (2×2)-block interaction model and is utilized in solving our problem.

DEFINITION 4. **(Graph Set Cover (GSC) Problem)** *Let U be the ground set and let $G = (V, E)$ be a graph, where each vertex and each edge is associated with a collection of elements in U: for a vertex v, let $S(v)$ be the elements associated with $v \in V$; for an edge (u, v), let $S(u, v)$ be the elements associated with $(u, v) \in E$ $(S(v), S(u, v) \subseteq U)$. Given a set of vertices $V_s \subseteq V$, let $G[V_s] = (V_s, E_s)$ be the subgraph induced by vertices V_s, and let $S(G[V_s])$ be the elements covered by the induced subgraph $G[V_s]$, i.e.,*

$$S(G[V_s]) = \bigcup_{v \in V_s} S(v) \bigcup_{(u,v) \in E_s} S(u, v). \tag{2}$$

Given this, the GSC problem seeks the smallest number of vertices, such that their induced subgraph can cover all the elements in the ground set, i.e.,

$$\arg\min_{|V_s|} S(G[V_s]) = U. \tag{3}$$

Figure 3 shows an example of GSC, where the ground set $U = \{1, 2, \cdots, 10\}$, and both vertices and edges in the graphs associate with a subset of U. The induced subgraph of vertices a, c, and d can cover U and is the optimal solution of this GSC problem.

It is not hard to see that the GSC problem is a generalization of the classical set cover problem [10]. Indeed, if we consider only the vertices are assigned with elements and the edges are not, then, this problem is a simple set cover problem. This also immediately shows the NP-hardness of this problem.

THEOREM 2. *The GSC problem is NP-hard.*

This problem is more closely related to the recently proposed *set-cover-with-pairs problem* [15].

DEFINITION 5. **(Set-Cover-with-Pairs Problem)** *Let U be the ground set and let $S = \{1, \ldots, M\}$ be a set of objects. For every $\{i, j\} \subseteq S$, let $\mathcal{C}(i, j)$ be the collection of elements in U covered by the pair $\{i, j\}$. The objective of the set cover with pairs (SCP) problem is to find a subset $S' \subseteq S$ such that*

$$\mathcal{C}(S') = \bigcup_{\{i,j\} \subseteq S'} \mathcal{C}(i, j) = U$$

with a minimum number of objects.

Note that we consider each object in SCP and each vertex in the GSC problem to be unweighted. Both can be easily reformulated as weighted versions but that is beyond scope of this work. It is easy to see that the SCP problem is also a special case of the GSC problem. Here, the input of SCP is a complete graph where each edge associates with a subset of the ground set U (not vertices). As demonstrated in [15], the approximation solution for SCP is much harder than the classical set cover problem, where a logarithmic bound approximation is available. The best known approach for the SCP is the following greedy algorithm:

Let R be the set of objects already covered, where $R = \emptyset$ initially; then at each iteration, we select the minimum of these two choices: 1) a node i such that $\frac{1}{|\mathcal{C}(R \cup \{i\}) \setminus \mathcal{C}(R)|}$ is minimum; and 2) a pair of nodes i and j such that $\frac{2}{|\mathcal{C}(R \cup \{i,j\}) \setminus \mathcal{C}(R)|}$ is minimum. We repeat such iteration until all elements are covered ($R = U$).

This greedy algorithm has been proved to yield an $O(\sqrt{N \log N})$ approximation ratio, where N is the cardinality of the ground set $N = |U|$, for the cardinality SCP problem. Note that we can directly employ this algorithm to the GSC problem as follows. We add a virtual vertex v_0 in the GSC problem, and then we link each vertex in the original graph with this virtual vertex. After that, we also move the sets of elements associated with each vertex $S(v)$ to the corresponding new edge, i.e., $S(v, v_0)$. We also assume the cost of virtual vertex v_0 is zero. Thus, we transform our GSC problem into a SCP problem.

Given this, to solve the GSC problem, we can always *1) cover the virtual vertex at the first step since there is no cost associated with it; 2) employ the greedy algorithm for SCP to complete the covering process*. Clearly, the approximation bound achieved by this procedure is no worse than $O(\sqrt{N \log N})$ (considering the optimal first step is always given by covering the virtual vertex).

4. A TWO-STAGE APPROACH

In this section, we describe an overall approach based on the GSC problem to generate a minimal (2×2)-block interaction model. Since it is hard to find the overall optimal number of core blocks for a collection of frequent itemsets F_α, we consider a two-stage algorithm for this purpose. In the first stage, we seek a minimal number of blocks which use only the $\oplus$ operator to support the entire collection of frequent itemsets. In the second stage, we seek a minimal number of blocks which use only the $\ominus$ operator to represent the blocks discovered in the first stage. Before we describe each of the two stages in details (Subsection 4.1), we first discuss several simple properties which can help simplify the search for the (2×2) block interaction model.

We will first simplify our problem by reducing both the targeted frequent itemsets and by limiting the scope of candidate core blocks. First, let the targeted collection frequent itemsets be F_α. We make the following observations:

OBSERVATION 1. *Let CF_α be the set of all closed frequent itemsets with minimal support α (an itemset is closed if no superset has as high a support level, $CF_\alpha \subseteq F_\alpha$. For a set of core blocks $\mathcal{B}$, if it can support each closed itemset in CF_α under the (2×2)-block support expression, i.e., it is a (2×2) block interaction model for CF_α, then, it is a (2×2) block interaction model for F_α (and vice versa).*

This observation allows us to focus on only the closed frequent itemsets CF_α in searching for the (2×2) block interaction model.

OBSERVATION 2. *Let $\mathcal{B}$ be the set of core blocks of an (2×2)-block interaction model for the collection of closed frequent itemsets CF_α. Then, for any block $B_i = I_i \times T_i \in \mathcal{B}$, we can always expand its transaction set to be $T(I_i)$, where $T(I_i)$ includes all the transactions containing (supporting) I_i. We refer to block $I_i \times T(I_i)$ as a* **supporting block**. *We can further expand the itemset I_i of the supporting block such that $I_i \subseteq I_i'$ and $T(I_i') = T(I_i)$ where there is no other itemset I_i'' with $I_i' \subset I_i''$ and $T(I_i) = T(I_i') = T(I_i'')$. In other words, I_i' is a closed itemset, and we refer to the expanded block as a* **closed supporting block**.

This observation confirms that in any (2×2) block interaction model, each core block can be replaced by a *closed supporting block*. Thus, to simplify our search, the only candidate core blocks we need to consider are closed blocks. In addition, since one of the goals of this work is to reduce the collection of frequent itemsets to a very small number of them, we can further limit the candidate core blocks to those whose itemsets are closed frequent itemsets (CF_α).

Formally, let the set of all candidate core blocks be $CB = \{I \times T(I) | I \in CF_\alpha\}$. Given this, we would like to search $\mathcal{B} \subseteq CB$, such that with minimal $|\mathcal{B}|$, each closed frequent itemset can be supported or explained by a (2×2) block expression:

$$\arg\min_{|\mathcal{B}|} \mathcal{B} \models CF_\alpha, \mathcal{B} \subseteq CB$$

Note that if $B_1, B_2 \in CB$, then, $I(B_1 \ominus B_2) \in CF_\alpha^2$, where $CF_\alpha^2 = \{I_1 \cup I_2 | I_1, I_2 \in CF_\alpha\}$. This observation is used in the next subsection for discovering the core blocks of a (2×2)-block interaction model.

4.1 Vertical Union ($\oplus$) and Horizontal Union ($\ominus$) Decomposition

As mentioned earlier, in order to find the set of core blocks for a (2×2) block interaction model, we consider a two-stage algorithm, where in each stage, we utilize a single type of operator ($\ominus$ or $\oplus$) to *support* a collection of (frequent) itemsets. The rationale for this two-stage approach comes from the following observation. Basically, for any closed frequent itemsets $I \in CF_\alpha$, we need a (2×2)-block expression to support it: $(B_1 \ominus B_2) \oplus (B_3 \ominus B_4)$, where B_1, B_2, B_3 and B_4 are core blocks. The optimization goal

is to minimize the total number of unique core blocks. Since this is an NP-hard problem and the direct optimization is hard to achieve, we consider the following heuristics. If the total number of unique core blocks is small, then their combinations $B_1 \ominus B_2$ and $B_3 \ominus B_4$ which are used in the (2×2)-block expression in supporting each closed frequent itemset also tend to be small. This observation inspires us to divide the overall problem into two subproblems which can be solved independently: in the first stage, we seek a minimal number of blocks which use only the $\ominus$ operator to support the entire collection of frequent itemsets; in the second stage, we seek a minimal number of blocks which use only the $\oplus$ operator to support the itemsets of the blocks discovered in the first stage. Here, we describe these two stages in detail.

Stage 1 (Minimizing Vertical Union Decomposition): In the first stage, we seek a minimal number of blocks ($\mathcal{C}$) which use only the $\oplus$ operator to support the entire collection of closed frequent itemsets (CF_α). Those blocks $\mathcal{C}$ discovered in the first stage then will be decomposed using $\ominus$ operator in the second stage. Specifically, the goal of the first stage is as follows:

DEFINITION 6. *(**Subproblem 1: Minimal Vertical Union Decomposition Problem**) Given a collection of closed frequent itemset CF_α, we seek a small set of blocks, $\mathcal{C} = \{C_1, \cdots, C_m\}$, where $C_i = I_i \times T(I_i)$ and $I_i \in CF_\alpha^2$, such that each itemset $I \in CF_\alpha$ can be supported or explained by at most two blocks C_i and C_j in $\mathcal{C}$, $C_i \oplus C_j \models I$ with respect to accuracy level ϵ_1 ($\epsilon_1 \leq \epsilon$): $I \subseteq I(C_i \oplus C_j)$ and*

$$|T(C_i \oplus C_j)| \geq (1 - \epsilon_1) supp(I).$$

Now we transform Subproblem 1 into an instance of the GSC problem (Subsection 3.2). Let $U_1 = CF_\alpha$ be the ground set to be covered. Let $G_1 = (V_1, E_1)$ be the graph we will construct.

Vertex Set Construction: Each vertex in the graph G_1 corresponds to a candidate block in $\{I_i \times T(I_i) | I_i \in CF_\alpha^2\}$, i.e., $|V| = |CF_\alpha^2|$. For each vertex $v \in V_1$, let B_v be the corresponding candidate block ($B_v \in \{I_i \times T(I_i) | I_i \in CF_\alpha^2\}$). Each vertex v is assigned with a set $S(v)$ which contains contains all the closed frequent itemsets being supported or explained by B_v:

$$S(v) = \{I \in CF_\alpha | B_v \models I\}$$

i.e., $I(B_v) \subseteq I$ and $|T(B_v)| = supp(I(B_v)) \geq (1 - \epsilon_1) supp(I)$.

Edge Set Construction: For any two vertices v_1 and v_2 in G_1, let set $S(v_1, v_2)$ include all the closed frequent itemsets which are supported by $B_{v_1} \oplus B_{v_2}$ but are not supported by B_{v_1} or B_{v_2}:

$$S(v_1, v_2) = \{I \in CF_\alpha | B_{v_1} \oplus B_{v_2} \models I\} \backslash (S(v_1) \cup S(v_2)) \quad (*)$$

i.e., $I(B_{v_1}) \cap I(B_{v_2}) \supseteq I$ and $|T(B_{v_1} \oplus B_{v_2})| \geq (1 - \epsilon_1) supp(I)$ (how to efficiently compute $|T(B_{v_1} \oplus B_{v_2})|$ will be discussed in Subsection 5.1). If $S(v_1, v_2)$ is not empty, then, these two vertices v_1 and v_2 are linked and set $S(v_1, v_2)$ is assigned to the corresponding edge.

It is easy to see that each subset of vertices V_s in G_1 which covers the ground set, i.e., $S(G_1[V_s]) = U_1$, corresponds to a set of blocks $\mathcal{C} = \{B_v | v \in V_s\}$ which supports the collection of closed frequent itemsets CF_α, i.e., $\mathcal{C} \models CF_\alpha$ (a solution for subproblem 1). Further, the optimal solution for the graph set problem is also the optimal solution for subproblem 1. Finally, we note that based on the greedy algorithm described in subsection 3.2, we obtain an approximation bound $O(\sqrt{N \log N})$ with $N = |CF_\alpha|$.

Stage 2 (Minimizing Horizontal Union Decomposition): In the second stage, we will seek a minimal number of blocks ($\mathcal{B}$) to support the blocks ($\mathcal{C}$) discovered in the first stage. Formally, the goal of this stage is formally described as follows.

DEFINITION 7. *(**Subproblem 2: Minimal Horizontal Union Decomposition Problem**) Let $\mathcal{C}$ be the set of blocks discovered in the first stage, we seek a minimal number of closed supporting blocks, $\mathcal{B} = \{B_1, \cdots, B_k\}$, where $B_i = I_i \times T(I_i), I_i \in CF_\alpha$, such that the itemset $I(C)$ of each block $C \in \mathcal{C}$ is supported or explained by at most two blocks B_i and B_j in $\mathcal{B}$, $B_i \ominus B_j \models I(C)$ with respect to accuracy level $\epsilon_2 = (\epsilon - \epsilon_1)/2$, i.e., $I(C) \subseteq I(B_i \ominus B_j)$ and*

$$|T(B_i \ominus B_j)| \geq (1 - \epsilon_2) supp(I(C)).$$

Now we transform Subproblem 2 into an instance of the GSC problem. Let $U_2 = \{I(C) | C \in \mathcal{C}\}$ be the ground set to be covered, where $\mathcal{C}$ is the collection of blocks generated in the first stage. Let $G_2 = (V_2, E_2)$ be the graph for this subproblem.

Vertex Set Construction: Each vertex v in G_2 corresponds to a closed supporting block $B = I \times T(I)$, where $I \in CF_\alpha$ and is assigned with a set $S(v)$, which contains all the frequent itemsets which are supported or explained by the corresponding block of v, denoted as B_v:

$$S(v) = \{I \in U_2 | B(v) \models I\}$$

i.e., $I(B_v) \supseteq I$ and $|T(B_v)| \geq (1 - \epsilon_2) supp(I)$.

Edge Set Construction: For any two vertices v_1 and v_2 in G_2, let set $S(v_1, v_2)$ include all the itemsets in U which are supported by $B_{v_1} \ominus B_{v_2}$, but are not supported by B_{v_1} or B_{v_2}:

$$S(v_1, v_2) = \{I \in U | B_{v_1} \ominus B_{v_2} \models I\} \backslash (S(v_1) \cup S(v_2)) \quad (**)$$

i.e., $I(B_{v_1}) \cup I(B_{v_2}) \supseteq I$ and $|T(B_{v_1} \ominus B_{v_2})| \geq (1 - \epsilon_2) supp(I)$. If $S(v_1, v_2)$ is not empty, then these two vertices v_1 and v_2 are linked and set $S(v_1, v_2)$ is assigned to the corresponding edge.

Again, we observe that each subset of vertices V_s in G_2 which covers the ground set, $S(G_2[V_s]) = U_2$, corresponds to a set of blocks $\mathcal{B} = \{B_v | v \in V_s\}$ which supports each itemset in $\mathcal{C}$, and the optimal solution for the graph set problem is also the optimal solution for subproblem 2. Using the greedy algorithm described in subsection 3.2, we can obtain an approximation bound $O(\sqrt{N \log N})$ with $N = |\mathcal{C}|$.

4.2 Correctness of the Two-Stage Approach

In this subsection, we prove that the blocks discovered in subproblems 1 and 2 form the core blocks of a (2×2) block interaction model for CF_α. Especially, we demonstrate how the user-preferred accuracy level ϵ is distributed correctly into the two stages, $\epsilon_1 \leq \epsilon$ to stage 1 and $\epsilon_2 = (\epsilon - \epsilon_2)/2$ to stage 2. The effect of different choices of ϵ_1 will be studied in Section 7. To facilitate our discussion, we first make the following observations for the cardinality of the transaction set generated by the vertical and horizontal union operators.

OBSERVATION 3. *Given any two blocks $B_1 = I_1 \times T(I_1)$ and $B_2 = I_2 \times T(I_2)$, we have $|T(I_1)| = supp(I_1)$ and $|T(I_2)| = supp(I_2)$. Then, the following properties hold:*

$$\begin{aligned} |B_1 \oplus B_2| &= |T(I_1) \cup T(I_2)| \\ &= |T(I_1)| + |T(I_2)| - |T(I_1) \cap T(I_2)| \\ &= |T(I_1)| + |T(I_2)| - |T(I_1 \cup I_2)| \\ &= supp(I_1) + supp(I_2) - supp(I_1 \cup I_2) \\ |B_1 \ominus B_2| &= |T_1 \cap T_2| = supp(I_1 \cup I_2) \end{aligned}$$

THEOREM 3. *The set of core blocks discovered by subproblems 1 and 2 form the basis for a (2×2) block interaction model for the collection of closed frequent itemsets CF_α.*

Proof sketch of Theorem 3 can be found in our technical report [17]

5. FAST ALGORITHM FOR CORE BLOCK DISCOVERY

In this section, we present the complete two-stage algorithm (Subsection 5.2) for discovering the core blocks of a (2×2)-block interaction model. In particular, we introduce several heuristics to deal with the scalability issues (Subsection 5.1).

5.1 Scalability Issues and Heuristics

As discussed in Section 4, our approach for discovering the minimal (2×2) block interaction model contains two stages: the vertical union decomposition and horizontal union decomposition stages. Each stage involves two major computational steps: 1) constructing the instance of a GSC problem and then 2) invoking the greedy GSC algorithm. However, both steps can be computationally expensive.

GSC construction: Let us look at the graph construction for the first stage. The vertex set V_1 corresponds to $CF_\alpha^2 = \{I_i \cup I_j | I_i, I_j \in CF_\alpha\}$ which includes the pairwise combination of any two closed frequent itemsets. To construct edge set E_1, we try to join any two vertices which involves $O(|CF_\alpha^2|^2)$ computational complexity. Even though $|CF_\alpha^2| << |CF_\alpha|^2$, this is still rather expensive.

A particular difficulty exists in computing $S(v_1, v_2)$, which needs to determine if $|T(B_{v_1} \oplus B_{v_2})| \geq (1-\epsilon_1)supp(I)$ for any frequent closed itemset I, where $B_{v_1} = I_1 \times T(I_1)$ and $B_{v_2} = I_2 \times T(I_2)$ $(I_1, I_2 \in CF_\alpha^2)$. We can utilize Observation 3 to solve this problem: $|T(B_{v_1} \oplus B_{v_2})| = supp(I_1) + supp(I_2) - supp(I_1 \cup I_2)$. Therefore, we need precompute the support for CF_α^2 and $CF_\alpha^4 = \{I_1 \cup I_2 | I_1, I_2 \in CF_\alpha^2\}$. Clearly, this can be very costly.

To deal with this problem, we employ two heuristics to effectively reduce the candidate block search space. The first heuristic considers the *support constraints* on the candidate blocks. If an itemset in CF_α^2 has lower support, then its likelihood to combine with other itemsets for recovering the supports of frequent itemsets may be smaller. To compensate, we may request each candidate itemset itself should cover at least one frequent itemset. This suggests that each itemset in CF_α^2 should have a support no less than $(1-\epsilon)\alpha$. In addition, since each candidate block can be improved by a corresponding *closed supporting block* (Observation 2), we further focus on only those closed itemsets in CF_α^2 instead of all the members. In sum, the set of candidate blocks in stage 1 is written as $CF_{(1-\epsilon)\alpha} \cap CF_\alpha^2$.

Interestingly, we note that graph construction for the second stage has a lesser scalability issue because its vertex set V_2 corresponds to CF_α, which is smaller than CF_α^2 in the first stage. In addition, only $|\mathcal{C}|$ blocks need to be covered, which can be orders of magnitude less than CF_α for the first stage.

GSC Greedy Algorithm: Now we take a look of the GSC algorithm based on the greedy algorithm for the set-cover-with-pairs (SCP) problem. In this greedy algorithm, we consider any pair of vertices in the graph as a candidate for covering. Actually, we only need to consider pairs of vertices that are adjacent (no improvement over single vertex if they are not connected), but this still results in $O(|E_1|)$ and $O(|E_2|)$ time complexity for each iteration of the GSC algorithm in the first and second stage, respectively.

It is interesting to observe the following properties on the GSC problem for the first stage.

LEMMA 2. *In graph $G_1 = (V_1, E_1)$ of the first stage, for any edge set $S(v_1, v_2)$, there is a vertex set $S(v_3)$, such that $S(v_1, v_2) \subseteq S(v_3)$.*

Proof: For edge (v_1, v_2) that covers any frequent itemsets in CF_α, then $|T(I_{v_1}) \cup T(I_{v_2})| \geq (1-\epsilon)\alpha$. Since our candidate blocks in the first stage consider at least $CF_{(1-\epsilon)\alpha} \cap CF_\alpha^2$, it means there is an itemset I_{v_3} such that $I_{v_3} \subseteq I_{v_1} \cap I_{v_2}$ and $supp(I_{v_3}) = supp(I_{v_1} \cap I_{v_2}) \geq |T(I_{v_1}) \cup T(I_{v_2})|$, i.e., I_{v_3} is the corresponding closed items of $I_{v_1} \cap I_{v_2}$. □

A similar observation can be made for $G_2 = (V_2, E_2)$ for the second stage. In these types of SCP (i.e., for each edge set $S(v_1, v_2)$, there is a vertex set $S(v_3)$ that covers at least as much), we observe that the ratio between the price of choosing an optimal vertex $\frac{1}{|\mathcal{C}(R \cup \{i\}) \setminus \mathcal{C}(R)|}$ and the price of choosing an optimal pair of vertices (an edge) $\frac{2}{|\mathcal{C}(R \cup \{i,j\}) \setminus \mathcal{C}(R)|}$ is no more than $3/2$ (Greedy algorithm for SCP, Section 3). This basically suggests that the benefit of choosing a pair (an edge) may be limited. Therefore, we can simplify and speed up the GSC algorithm as follows:

1. *Let R be the set of objects already covered, $R = \emptyset$ initially;*
2. *At each iteration, we select a vertex v such that $|S(G[R \cup \{i\}]) \setminus S(G[R])|$ is maximum;*
3. *We repeat 2) until all elements in U are covered.*

This algorithm is referred as *GraphSetCover*.

Algorithm 1 CoreBlockDiscovery(DB,CF_α, ϵ)

Require: CF_α: frequent closed itemsets in DB with minimum support α
Require: ϵ: user-defined accurracy level
{Stage 1: Block Vertical Union ($\oplus$) Decomposition}
$CF_\alpha^2 \leftarrow \{I_1 \cup I_2 | I_1, I_2 \in CF_\alpha\}$;
$CF \leftarrow CF_{(1-\epsilon)\alpha} \cap CF_\alpha^2$ {reducing the candidate blocks};
$CF^2 \leftarrow \{I_1 \cup I_2 | I_1, I_2 \in CF\}$;
ComputeSupport($CF^2 \backslash CF_{(1-\epsilon)\alpha}$); {Compute support for each itemset in CF}
{Vertex Set Construction:}
for all $I_v \in CF$ **do**
  $V_1 \leftarrow V_1 \cup \{(I_v, supp(I_v)\}$;
  $S(v) \leftarrow \{I \in CF_\alpha | I \subseteq I_v \wedge supp(I_v) \geq (1-\epsilon_1)supp(I)\}$;
end for
{Edge Set Construction:}
for all $(I_1, I_2) \in CF \times CF$ **do**
  $S(v_1, v_2) \leftarrow \{I \in CF_\alpha | I_1 \cap I_2 \supseteq I \wedge supp(I_1) + supp(I_2) - supp(I_1 \cup I_2) \geq (1-\epsilon_1)supp(I)\} \backslash (S(v_1) \cup S(v_2))$;
  if $S(v_1, v2) \neq \emptyset$ **then**
    $E_1 \leftarrow E_1 \cup \{(v_1, v_2)\}$;
  end if
end for
$\mathcal{C} \leftarrow$ GraphSetCover($G_1(V_1, E_1), CF_\alpha$);
{Stage 2: Block Horizontal Union ($\ominus$) Decomposition}
$U \leftarrow \{I(C) | C \in \mathcal{C}\}$;
{Vertex Set Construction:}
for all $I_v \in CF_\alpha$ **do**
  $V_2 \leftarrow V_2 \cup \{(I_v, supp(I_v)\}$;
  $S(v) \leftarrow \{I \in U | I \subseteq I_v \wedge supp(I_v) \geq (1-\epsilon_2)supp(I)\}$;
end for
{Edge Set Construction:}
for all $(I_1, I_2) \in CF_\alpha \times CF_\alpha$ **do**
  $S(v_1, v_2) \leftarrow \{I \in U | I_1 \cup I_2 \supseteq I \wedge supp(I_1 \cup I_2) \geq (1-\epsilon_2)supp(I)\} \backslash (S(v_1) \cup S(v_2))$;
  if $S(v_1, v2) \neq \emptyset$ **then**
    $E_2 \leftarrow E_2 \cup \{(v_1, v_2)\}$;
  end if
end for
$\mathcal{B} \leftarrow$ GraphSetCover($G_2(V_2, E_2), U$);

5.2 Algorithm Description

Algorithm 1 sketches the complete two-stage approach (including both vertical union ($\oplus$) and horizontal union ($\ominus$) decomposition, Section 4.1).

Lines 1 to 15 describe Stage 1 of Algorithm 1 for vertical union decomposition. We first generate the candidate itemsets and their supports in Lines 1 to 4. Note that to compute the support for candidate itemsets in $CF^2 \backslash CF_{(1-\epsilon)\alpha}$, we simply organize those itemsets in a prefix tree and then count their occurrences by enumerating the subsets in each transaction. We refer to this procedure as *ComputeSupport* (details omitted). We then construct the vertex set of G_1 (Lines 5 to 8) and its edge set (Lines 9 to 13. Once the graph G_1 is constructed, the GSC algorithm is invoked to cover CF_α using core blocks in $\mathcal{C}$.

Similarly, Stage 2 of Algorithm 1 for horizontal union decomposition is described from Lines 16 to 27. Here, the ground set U (to be covered) includes all the itemsets of blocks generated by Stage 1. We then construct the vertex set and edge set of G_2 (Lines 17 to 25) Finally, after the graph G_2 is constructed, the GSC algorithm is called to find the final core blocks $\mathcal{B}$, which is also the core blocks of the entire (2×2)-block interaction model (Line 27).

Computational Complexity: Algorithm 1 consists of four parts: (1) vertex construction of G_1, (2) edge construction of G_1, (3) vertex construction of G_2, and (4) edge construction of G_2, plus graph set cover for G_1 and G_2. The total time complexity for the four parts is $O(|CF|f(|CF_\alpha|))+O(|CF|^2|f(CF_\alpha|))+O(|CF_\alpha|f(|U|))$ $+O(|CF_\alpha|^2 f(|U|)) = O(|CF|^2 f(|CF_\alpha|) + |CF_\alpha|^2 f(|CF|))$. Here $f()$ is the cost of function to check if any itemset in CF_α is included in a vertex itemset or edge itemset. The naïve implementation based on a linear scan yields $f(|CF_\alpha|) = |CF_\alpha|$. However, we can improve this procedure by treating CF_α as a transactional database where each transaction is an itemset. Then, we can organize it by the vertical format, i.e., for each item, which transactions contains it. Thus, the checking process can be improved significantly. The time complexity of GSC may vary according to its detailed implementation. We only need to consider Stage 1 as it is more expensive than Stage 2. A simple greedy algorithm has a time complexity proportional to the size of the the graph (in our case $O(|CF|)$) and the set of the ground set (in our case $O(CF_\alpha)$) to be covered. Putting these together, the overall time complexity of Algorithm 1 is $O(|CF|^2 f(|CF_\alpha|) + |CF_\alpha|^2 f(|CF|))$.

6. RELATED WORKS

Numerous efforts have been made to reduce the number of frequent patterns, especially for itemsets. Well-known examples include maximal frequent patterns [19], closed frequent patterns [21], free-sets [5], disjunction-free sets [7], non-derivable itemsets [9, 8, 20], and δ-cluster notation [29]. Most of these methods try to identify certain rules to filter out "non-essential" itemsets. Even though some of the rules utilize disjunctive rules or deductive rules which share a certain spirit with our block-interaction modeling, their objectives do not address utilizing a small collection of basic itemsets to "generatively" explain or recover other frequent itemsets. In our scheme, even if an itemset can be explained by other itemsets, it may still serve as a core block. We note that δ-cluster also tries to apply the set cover idea to concisely represent the collection of frequent itemsets. However, it does not provide a generative view. Indeed, the δ-cluster method can be viewed as a special case of ours: a (1×1)-block interaction model with no block union operators.

In [18], we developed an approach to provide a generative view of an entire collection of itemsets. However, that model cannot recover support information for an individual itemset. Also, the optimization goal of [18] is not to reduce the number of frequent itemsets to a small core set, but to minimize the representation or storage cost of maximal itemsets. In [18], we proposed a bipartite graph set cover problem which is equivalent to a set cover problem. In this work, the proposed *graph set cover* problem is much more general and more difficult than the classical set cover problem.

Datasets	$\mathcal{I}$	$\mathcal{T}$	density
connect	129	67557	dense
pumsb	7116	49046	dense
chess	75	3,196	dense
retail	16469	88162	sparse
T40I10D100K	1000	100000	sparse

Table 1: Datasets Characters. $\mathcal{I}$ is the total number of items and $\mathcal{T}$ is the total number of transactions

7. EXPERIMENTAL RESULTS

In this section, we report the effectiveness of (2×2)-block interaction modeling in summarizing the frequent itemsets on both real and synthetic datasets. Specifically, we are interested in the following questions:

1. How does our block interaction model(**B.I.**) compare with the state-of-art summarization schemes, including Maximal Frequent Itemsets [19](**MFI**), Close Frequent Itemsets [21](**CFI**), Non-Derivable Frequent Itemsets [9](**NDI**), and Representative pattern [29](**δ-Cluster**).
2. How do different parameters, including α, ϵ, and ϵ_1 affect the conciseness of the block modeling, i.e., the number of core blocks?

In order to answer the above questions, we design three groups of experiments:

Group 1: In the first group of experiments, we vary the support level α for each dataset with a fixed user-preferred accuracy level ϵ (either 5% or 10%) and fix $\epsilon_1 = \frac{\epsilon}{2}$.

Group 2: In the second group of experiments, we study how user-preferred accuracy level ϵ would affect the model conciseness (the number of core blocks). Here, we vary ϵ generally in the range from 0.1 to 0.2 with a fixed support level α and $\epsilon_1 = \frac{\epsilon}{2}$.

Group 3: In the third group of experiments, we study how the distribution of accuracy level ϵ_1 in the two stages would affect the model conciseness. We vary ϵ_1 between 0.1ϵ and 0.9ϵ with fixed support level α and the overall accuracy level ϵ.

Datasets and Experimental Environment: We conducted our evaluation on a total of 5 datasets, including four real datasets and one synthetic dataset. Three of them are dense datasets and two of them are sparse datasets. All of them are publicly available on the FIMI repository [1]. The basic characteristics of the datasets are listed in Table 1. In our experiments, we use the MAFIA algorithm [6], which is publicly available online[2], to generate frequent itemsets, closed frequent itemsets and maximal frequent itemsets. Our algorithms were implemented in C++ and run on Linux 2.6 on an Intel Xeon 3.2 GHz processor with 4GB memory.

Results from Group 1: Table 2, 3, 4, and 5 show the results of Group 1 on the real datasets *connect*, *pumsb*, *chess*, and *retail*, respectively. Table 6 shows the results on the synthetic dataset *T40I10D100K*. In these experiments, we can see clearly that the (2×2) interaction model consistently produces the most concise representation for different support levels. In particular, our results show that the number of core blocks needed to represent the entire collection of frequent itemsets, including their support information, is even less than the number of maximal frequent itemsets (MFI), one of the best summarization methods for frequent itemsets, but which does not include support information.

[1] http://fimi.cs.helsinki.fi/data/

[2] http://himalaya-tools.sourceforge.net/Mafia/

α	MFI	CFI	NDI	δ-Cluster	B.I.
0.92	175	2212	168	178	56
0.91	192	2819	184	196	56
0.90	222	3486	199	222	72
0.89	261	4218	223	279	85
0.88	313	5106	240	332	89

Table 2: Group1.Connect: $\epsilon = 0.05$

α	MFI	CFI	NDI	δ-Cluster	B.I.
0.90	259	1465	585	259	48
0.89	348	2186	763	348	82
0.88	500	3160	501	988	88
0.87	633	4508	1200	634	99
0.86	825	6245	1470	826	262

Table 3: Group1.Pumsb: $\epsilon = 0.1$

α	MFI	CFI	NDI	δ-Cluster	B.I.
0.875	74	1059	133	83	81
0.850	119	1885	172	137	82
0.825	176	3189	218	209	126
0.800	226	5083	281	288	109
0.775	325	7679	352	426	266

Table 4: Group1.Chess: $\epsilon = 0.05$

α	MFI	CFI	NDI	δ-Cluster	B.I.
0.007	167	315	317	294	136
0.006	219	417	418	391	176
0.005	284	580	582	545	241
0.004	424	831	838	783	335
0.003	692	1393	1410	1325	538

Table 5: Group1.Retail: $\epsilon = 0.05$

α	MFI	CFI	NDI	δ-Cluster	B.I.
0.032	608	685	686	685	458
0.031	645	730	731	730	472
0.030	700	793	794	793	486
0.029	741	842	843	842	495
0.028	812	924	925	924	506

Table 6: Group1.T40I10D100K: $\epsilon = 0.1$

ϵ	MFI	CFI	NDI	δ-Cluster	B.I.
0.06	222	3486	199	225	104
0.08	222	3486	199	223	50
0.1	222	3486	199	222	40
0.12	222	3486	199	222	27
0.14	222	3486	199	222	19

Table 7: Group2.Connect: $\alpha = 0.9$

ϵ	MFI	CFI	NDI	δ-Cluster	B.I.
0.06	219	417	418	390	176
0.07	219	417	418	389	175
0.08	219	417	418	389	175
0.09	219	417	418	220	233
0.1	219	417	418	389	203

Table 8: Group2.Retail: $\alpha = 0.006$

ϵ_1	MFI	CFI	NDI	δ-Cluster	B.I.
0.01	259	1465	585	259	28
0.03	259	1465	585	259	39
0.05	259	1465	585	259	48
0.07	259	1465	585	259	87
0.09	259	1465	585	259	258

Table 9: Group3.Pumsb: $\alpha = 0.9$**,** $\epsilon = 0.1$

ϵ_1	MFI	CFI	NDI	δ-Cluster	B.I.
0.005	219	417	418	391	216
0.015	219	417	418	391	401
0.025	219	417	418	391	176
0.035	219	417	418	391	175
0.045	219	417	418	391	175

Table 10: Group3.Retail: $\alpha = 0.006$**,** $\epsilon = 0.05$

Specifically, Table 2 shows the number of core blocks (core itemsets) in B.I. is on average more than 3 times smaller than the number of patterns in MFI, NDI and δ-Cluster. It is 50 times more compact than CFI. Table 3 shows that B.I. is 5 times better than MFI and δ-Cluster, 9 and 32 times better than CFI and NDI respectively. In Table 4, the result sizes of MFI, NDI and δ-Cluster are around 1.5 times more than the number of core blocks, and B.I. is 27 times better than CFI. In Table 5, the block size in B.I. is noticeably smaller than MFI, and less than half of those results produced by other methods, while in Table 6, our B.I. method achieves 1.5 times better compactness than the others.

Results from Group 2: In the second group of experiments, we fix α for each dataset and vary ϵ between 1 % and 20%. Due to the lack of space, we only provide the experimental results on two datasets, one dense dataset (Connect) and one sparse dataset (Retail). For the dense dataset *connect*, Tahle 7 shows the number of core blocks shrinks as the user-preferred accuracy level grows. Clearly, this is consistent with the intuition that less accurate recovery needs fewer blocks and more accurate recovery needs more core blocks. Specifically, for the dense dataset (Connect), the number of core blocks (itemsets) is up to more than 10 times better than MFI, NDI, and δ-Clusters, and on average 183 times smaller than CFI. Our method also works well for the sparse dataset: Table 8 shows that the number of core blocks is smaller than that of MFI, and 2 times more compact than CFI, NDI and δ-Cluster.

Results from Group 3: For the similar reason, we only report on two datasets for the third group of experiments. We first fix α to be 0.9 for the dense dataset *Pumsb*, 0.006 for the sparse dataset *Retail*, and vary ϵ_1 from 0.1ϵ to 0.9ϵ.

For the dense dataset *Pumsb*, when $\epsilon_1 = 0.7\epsilon$, B.I. is about 3 times better than MFI and δ-Cluster, 6.7 times better than NDI, and 16.8 times better than CFI. However, when ϵ_1 decreases to 0.1ϵ, B.I. performs almost 10 times better than MFI and δ-Cluster, 20 times better than NDI, and even 52 times better than CFI. Interestingly, we can observe quite different performance of B.I. on the sparse dataset *Retail* (Table 10) when ϵ_1 decreases. This suggests that ϵ_1 has very different impacts on B.I. in different datasets. While for a given dataset, we may be able to obtain a general understanding of its performance (by looking at how each stage can effectively explain their input itemsets), an analytical method for optimizing ϵ_1 for different dataset is an interesting open question.

8. CASE STUDY

In this section, we study the effectiveness of applying block interaction model on a real biomedical application. First, we apply our block interaction model on the human phenotype-to-gene dataset [3] with 1.5% support level. After discarding blocks which contain too few genes to be biologically significant, we obtain 63 blocks containing 63 gene clusters for the following study, which suggests these core blocks may contain very promising biomedical information.

The identified gene clusters may contain information regarding

[3]http://www.human-phenotype-ontology.org/index.php/downloads.html

diseases groups and may also be used as new disease related gene groups or even biomarkers for prognosis. To test this, we selected a dataset of gene expression profiles (microarray) for 295 breast cancer patients (the NKI dataset [24, 23]). This cohort contains patients with multiple subtypes of breast cancers including lymph node (LN) positive, LN-negative, estrogen receptor (ER) positive and ER-negative. Previously there have been many studies for deriving prognostic gene markers for breast cancers using this dataset. In this paper, we focus on one subtype with poor prognosis (e.g., metastatic, short expected survival time, and poor response to drugs), namely the ER-negative. Despite the fact that this subtype affects about 1/3 of breast cancer patients, there has not been much studies on the molecular signature for predicting the prognosis of this group of patients. For each gene cluster, we use them as features to separate selected subtypes of patients into two subgroups using K-means algorithm (K = 2, repeat 100 times for each test) based the expression profiles of these patients over the feature genes. Note that in a cluster only genes whose names exactly appear in the NKI dataset will take effects in our test. The survival curves (Kaplan-Meier curves) are then plotted for each subgroup and the log-rank test is used to examine if there is significant difference in survival times between the two subgroups which indicates that the gene cluster may be used as a biomarker for predicting patient prognosis in breast cancer. We also compare our results with the well-known 70-gene signature derived in [24].

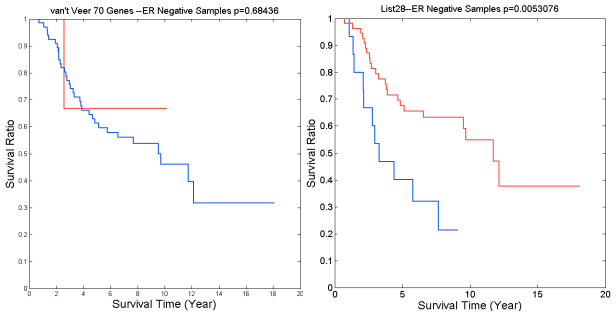


Figure 4: Left: The Kaplan-Meier curves for the two groups from the ER-negative patients separated using the cluster generated from by the 70-gene signature. Right: The Kaplan-Meier curves for the same group of patients obtained using the BI algorithm.

For the ER-negative patients, out of the 63 gene lists, 19 of them can separate the patients into two groups with significant difference in survival times ($p < 0.05$). For example, one gene cluster (FGFR2, PEX10, PEX14, PEX13, PEX1, PEX26, PEX3, PEX19, PEX5) show separation of the patients with significant difference in survival time ($p = 0.00531$). As a comparison, the well-known 70-gene signature cannot even be applied to this group of patients. Our result not only presents a potential new biomarker for the ER-negative breast cancer, but also provides a new biological hypothesis on the role of peroxisome in the ER-negative breast cancer progression since this cluster of genes are highly enriched with peroxisomal genes (Fisher's exact test for Gene Ontology enrichment analysis $p < 10^{-22}$).

9. CONCLUSIONS

Can we summarize a collection of frequent itemsets and provide accurate support information using only a small number of frequent itemsets? This problem is not only of practical importance, but also of theoretical interest. It has become one of the central problems in recent frequent pattern mining research. The existing methods have mainly focused on leveraging simple and basic rules to define and then remove non-essential patterns. In this work, we take a different route by focusing on different but also fundamentally important questions: *How does the complexity of frequent patterns arise? Can the large number of frequent itemsets be generated from a small number of patterns through their interactions?* In our search for a generative view of the collection of frequent patterns, we have developed the novel *block-interaction* model to concisely summarize a collection of frequent patterns. This model not only brings a new view of the frequent patterns, but also opens a set of new research questions: Should frequent patterns be a phenomenon or the targeted knowledge? What is the underlying knowledge/rules for frequent patterns? Will such knowledge (like the core block/itemsets) be more useful than the frequent pattern themselves? In addition, this model also addresses an important challenge for data mining: Given a set of mining results, how can we explain them or visualize them to the end users? The model developed here clearly gears towards such a goal. Finally, our study also opens up a list of research questions for algorithmic research, for instance, can we develop an algorithm for graph set cover with better approximation bound?

10. REFERENCES

[1] Foto Afrati, Aristides Gionis, and Heikki Mannila. Approximating a collection of frequent sets. In *KDD*, pages 12–19, 2004.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD Conference*, pages 207–216, May 1993.

[3] 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*, pages 487–499, 1994.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, 1995.

[5] Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti. Free-sets: A condensed representation of boolean data for the approximation of frequency queries. *Data Min. Knowl. Discov.*, 7(1):5–22, 2003.

[6] Douglas Burdick, Manuel Calimlim, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Mafia: A maximal frequent itemset algorithm. *IEEE Trans. Knowl. Data Eng.*, 17(11):1490–1504, 2005.

[7] Artur Bykowski and Christophe Rigotti. Dbc: a condensed representation of frequent patterns for efficient mining. *Inf. Syst.*, 28(8):949–977, 2003.

[8] T. Calders, C. Rigotti, and J-F. Boulicaut. A survey on condensed representations for frequent sets. In J-F. Boulicaut, L. de Raedt, and H. Mannila, editors, *Constraint-Based Mining*, volume 3848 of *LNCS*. Springer, 2006.

[9] Toon Calders and Bart Goethals. Non-derivable itemset mining. *Data Min. Knowl. Discov.*, 14(1):171–206, 2007.

[10] Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[11] Floris Geerts, Bart Goethals, and Taneli Mielikäinen. Tiling databases. In *Discovery Science*, pages 278–289, 2004.

[12] Aristides Gionis, Heikki Mannila, and Jouni K. Seppänen.

Geometric and combinatorial tiles in 0-1 data. In *PKDD*, pages 173–184, 2004.

[13] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Min. Knowl. Discov.*, 15(1):55–86, 2007.

[14] Jiawei Han, Jianyong Wang, Ying Lu, and Petre Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *ICDM*, pages 211–218, 2002.

[15] Refael Hassin and Danny Segev. Proceedings of the 25th annual conference on foundations of software technology and theoretical computer science (fsttcs). In *FSTTCS*, pages 164–176, 2005.

[16] Ruoming Jin, Muad Abu-Ata, Yang Xiang, and Ning Ruan. Effective and efficient itemset pattern summarization: regression-based approaches. In *KDD*, pages 399–407, 2008.

[17] Ruoming Jin, Yang Xiang, and Hui Hong. Block interaction: A generative summation scheme for frequent patterns. Technical Report TR-KSU-CS-2010-02, Computer Science, Kent State University, May 2010.

[18] Ruoming Jin, Yang Xiang, and Lin Liu. Cartesian contour: a concise representation for a collection of frequent sets. In *KDD*, pages 417–426, 2009.

[19] Roberto J. Bayardo Jr. Efficiently mining long patterns from databases. In *SIGMOD Conference*, pages 85–93, 1998.

[20] Juho Muhonen and Hannu Toivonen. Closed non-derivable itemsets. In *PKDD*, pages 601–608, 2006.

[21] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT*, pages 398–416, 1999.

[22] Ardian Kristanto Poernomo and Vivekanand Gopalkrishnan. Cp-summary: a concise representation for browsing frequent itemsets. In *KDD*, pages 687–696, 2009.

[23] Marc J. van de Vijver, Yudong D. He, Laura J. van 't Veer, Hongyue Dai, Augustinus A.M. Hart, Dorien W. Voskuil, George J. Schreiber, Johannes L. Peterse, Chris Roberts, Matthew J. Marton, Mark Parrish, Douwe Atsma, Anke Witteveen, Annuska Glas, Leonie Delahaye, Tony van der Velde, Harry Bartelink, Sjoerd Rodenhuis, Emiel T. Rutgers, Stephen H. Friend, and René Bernards. A gene-expression signature as a predictor of survival in breast cancer. *The New England Journal of Medicine*, 347(25):1999–2009, 2002.

[24] Laura J. van 't Veer, Hongyue Dai, Marc J. van de Vijver, Yudong D. He, Augustinus A. M. Hart, Mao Mao, Hans L. Peterse, Karin van der Kooy, Matthew J. Marton, Anke T. Witteveen, George J. Schreiber, Ron M. Kerkhoven, Chris Roberts, Peter S. Linsley, RenÍę Bernards, and Stephen H. Friend. Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415(6871):530–536, 2002.

[25] Chao Wang and Srinivasan Parthasarathy. Summarizing itemset patterns using probabilistic models. In *KDD*, pages 730–735, 2006.

[26] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, 2003.

[27] Yang Xiang, Ruoming Jin, David Fuhry, and Feodor F. Dragan. Succinct summarization of transactional databases: an overlapped hyperrectangle scheme. In *KDD*, pages 758–766, 2008.

[28] Dong Xin, Hong Cheng, Xifeng Yan, and Jiawei Han. Extracting redundancy-aware top-k patterns. In *KDD*, 2006.

[29] Dong Xin, Jiawei Han, Xifeng Yan, and Hong Cheng. Mining compressed frequent-pattern sets. In *VLDB*, 2005.

[30] Xifeng Yan, Hong Cheng, Jiawei Han, and Dong Xin. Summarizing itemset patterns: a profile-based approach. In *KDD*, 2005.

[31] M. T. Yang, R. Kasturi, and A. Sivasubramaniam. An Automatic Scheduler for Real-Time Vision Applications. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, 2001.

[32] Mohammed J. Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2002.

Authorship Classification: A Syntactic Tree Mining Approach *

Sangkyum Kim, Hyungsul Kim, Tim Weninger, Jiawei Han
University of Illinois at Urbana-Champaign
{kim71, hkim21, weninge1, hanj}@illinois.edu

ABSTRACT

In the past, there have been dozens of studies on automatic authorship classification, and many of these studies concluded that the writing style is one of the best indicators of original authorship. From among the hundreds of features which were developed, *syntactic features* were best able to reflect an author's writing style. However, due to the high computational complexity of extracting and computing syntactic features, only simple variations of basic syntactic features of function words and *part-of-speech* tags were considered. In this paper, we propose a novel approach to mining discriminative k-embedded-edge subtree patterns from a given set of syntactic trees that reduces the computational burden of using complex syntactic structures as a feature set. This method is shown to increase the classification accuracy. We also design a new kernel based on these features. Comprehensive experiments on real datasets of news articles and movie reviews demonstrate that our approach is reliable and more accurate than previous studies.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text Analysis*; H.3.3 [**Information Search and Retrieval**]: Clustering; H.2.8 [**Database Applications**]: Data Mining

General Terms

Algorithms, Pattern

*This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (award number OCI 07-25070) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign, its National Center for Supercomputing Applications, IBM, and the Great Lakes Consortium for Petascale Computation. This work was also sponsored in part by the National Science Foundation (under grants IIS-09-05215, CCF-0905014, and CNS-0931975) and an NDSEG Fellowship award. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies.

UP'10, July 25th, 2010, Washington, DC, USA.

Keywords

Authorship Classification, Text Mining, Text Categorization, Discriminative Pattern, Closed Pattern

1. INTRODUCTION

In computational linguistics and text mining domains, there have been three classical classification problems: topic classification, genre classification, and authorship classification. Among those three problems, the most difficult one is encountered when we try to classify documents in terms of their authorship (known as *authorship classification*, authorship attribution and/or authorship discrimination). This problem can be thought of as classifying documents based on the writing styles of the authors. It is a nontrivial problem even for the human beings: while a human can easily identify the topic and genre of a document, identifying its authorship is harder. Even worse, if the documents are from the same topic and genre, the task becomes much harder.

In the era of excessive electronic texts, authorship classification has been more and more important with a wide variety of applications. Besides the early works of analyzing disputed plays of *Shakespeare*(1887) [20] or anonymous documents of *The Federalist Papers*(1964) [23], it could also be used to identify authors of short 'for sale' messages in a newsgroup [37] and even for forensic investigations by identifying authorship of e-mail messages [2]. Detecting plagiarism or copyright infringement of unauthorized reuse of source code by establishing a profile of an author's style is another important application of authorship classification [6].

Existing approaches of authorship classification use various methods to extract effective features, most commonly style markers such as function words [11, 35, 1, 13] and grammatical elements such as part of speech (*POS*) tags [3, 12, 36]. Function words are the most common words that have little semantic content of their own but usually indicate a grammatical relationship or generic property. The success of using function words and *POS* tags as features for authorship classification indicates the usefulness of syntactic information.

Unfortunately, research on more complex syntactic structures has not yet been flourished because of the lack of a reliable, automatic tool which retrieves syntactic structures, and because of the high computational cost associated with syntactic structure-based algorithms. Instead, several rather simple syntactic structures, such as rewrite rules [3, 12] and n-grams of *POS* tags [11, 12, 15, 14] were discussed.

Lately, several advanced techniques were developed which greatly improved the performance of *Natural Language Pro-*

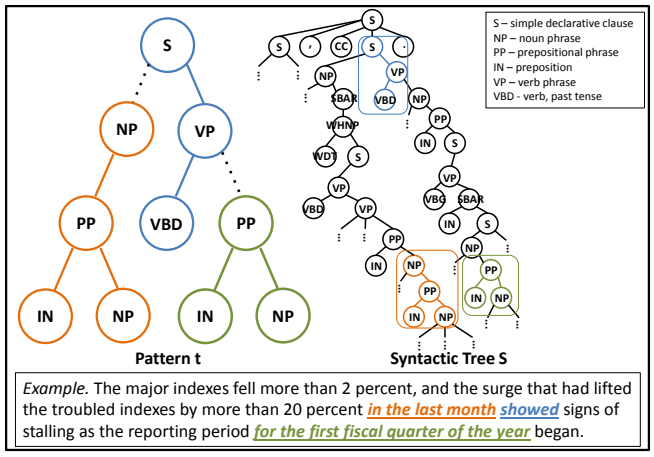


Figure 1: A 2-*ee* subtree pattern t is mined from two *NY Times* journalists Jack Healy and Eric Dash who worked in the same business department. On average, 21.2% of Jack's sentences contained t while only 7.2% of Eric's sentences contained t.

cessing(NLP) tools [18] enabling reliable, highly accurate sentence parsing into a syntactic tree of POS tags. Recently, emerging research of question answering (QA) systems [22, 4, 30] adapted these advanced preprocessing techniques to develop a tree kernel function that computed a matching score of two syntactic trees in order to retrieve similar questions or to classify questions in the QA system. But those approaches have several problems to be applied to authorship classification problem for the following reasons: (i) Even though their tree kernel utilizes more complex features than earlier works of rewrite rules and n-grams of POS tags, those features were in restricted forms of subtrees of a syntactic tree. There is a need to design a feature set that can capture syntactic information of a longer and more complicated sentence structure than simple question formats. (ii) The number of features becomes explosive once we consider all possible subtrees (even with some restrictions), and it leads to a burden on computational cost, however efficient a kernel computation is. (iii) Existing tree kernels work only for a data set of trees, not a data set of sets of trees. A question can be transformed into a syntactic tree, but a document which consists of a set of sentences becomes a set of syntactic trees.

In this paper, we propose a novel syntactic feature set of tree fragments allowing at most k-embedded edges (in short, k-*ee* subtree). Compared with previous feature sets that consists of distinct subtree components, our new feature set captures the relationship between $k+1$ subtree components of a syntactic tree, which leads to a better representation of a data set of long and complex sentences. To reduce the number of features, we only mine a set of discriminative and frequent k-*ee* subtrees, which results in higher accuracy by avoiding overfitting to the training data and by not generating non-discriminative features that often deteriorate the performance. For the classification, we introduce a new tree kernel by defining a proper value for each corresponding feature to be well-defined and effective on a data set of sets of trees.

Figure 1 gives an example of a k-*ee* subtree pattern t for $k = 2$. Pattern t is composed of three smaller induced subtrees, which are connected by two embedded edges (S,NP) and (VP,PP). The differences of pattern distributions between two authors suggest that a set of k-*ee* subtree patterns can be utilized as a good feature set for authorship classification.

Our framework can also be considered as a tree-kernel method, but it is different from previous tree-kernel approaches of a QA system in the following ways: First, our objects to be classified are in a more general form. Previous tree-kernel methods work for questions where each question becomes one syntactic tree, while our approach are based on documents where each document is a set of syntactic trees. Since previous tree kernels work only between two syntactic trees not between two set of syntactic trees, it cannot be directly applied to the authorship classification problem. Second, we use a more general feature set of subtree patterns allowing k-embedded edges, which works well to represent long and complex syntactic structures of a sentence. Third, a tree-kernel method essentially matches two trees without looking at the entire dataset. That is, it counts the common number of subtree patterns of two syntactic trees. But, our approach can get an overview of different classes of training data to select the discriminative patterns as features.

We adapt the framework of discriminative frequent pattern mining which showed good results for various problem settings in unstructured and semi-structured data mining such as mining discriminative frequent itemset, sequence, and graph patterns to classify UCI datasets, software behaviors, and chemical compound data, respectively [8, 19, 31].

While other syntactic features utilize the *bag-of-words* model to represent a document – which assigns the number of occurrences of a feature to its value – k-*ee* subtree patterns cannot adapt the same way due to the overlapped occurrences. Since we consider all subtrees and even allow k-embedded edges, a huge number of occurrences might overlap each other which would lead to an exaggeration of a feature value. At the other end, binary features will lose most of their occurrence information which results in either 0 or 1. Therefore, we design a new way to assign proper values for k-*ee* subtree features in between two extreme ends.

To validate the utility of our new feature set to others, for fair comparisons, we apply the same classification algorithm (SVM) to various feature sets over several real datasets. Experimental results demonstrate the effectiveness of our newly proposed feature set of k-*ee* subtree patterns over the well-known existing feature sets.

In summary, the contributions of this paper are as follows:

- We propose a new feature set of k-*ee* subtree patterns for authorship classification.
- We develop an algorithm to mine discriminative k-*ee* subtree patterns.
- We propose a new document representation based on our new feature set of k-*ee* subtree patterns that is proper for data of sets of trees.
- Through experiments on various datasets, we demonstrate the utility of our proposed framework to provide an effective solution for the authorship classification problem.

The rest of the paper is organized as follows. Section 2 presents an overview of the related works. In Section 3, we introduce various preliminary concepts, define our new feature *k-ee* subtree pattern, and describe our *k-ee* subtree pattern-based authorship classification framework. Section 4 presents a closed and frequent *k-ee* subtree mining algorithm with several pruning techniques. In Section 5, we explain discriminative pattern mining with a sequential coverage approach. We report our experimental results in Section 6, followed by conclusions and future work in Section 7.

2. RELATED WORKS

There are two main steps involved in any authorship classification algorithms: feature extraction step and classification step based on extracted features. For the feature extraction step, since the earliest works that used a small number of common words such as '*and*', '*to*' as a feature set, nearly 1,000 different features have been studied including sentence length, chi-square score, lexical richness [25, 17], vocabulary richness [10], function words [1], word n-grams [26], character n-grams [14], and rewrite rules [3] with lots of controversy on their effectiveness. Even though there was an issue of fair comparison between feature sets because previous works conducted experiments based on their own data sets with different classification methods [35, 27], function words and rewrite rules were considered to show reliable results. In [27], comprehensive survey on different feature sets were presented.

For the classification step, even though lots of new features were explored for authorship classification, most of the classification algorithms were simply adapted from well-known classification algorithms in other domains such as *PCA* [16], *k*-nearest neighbor, decision tree, bayesian networks [35], language model [36], and *SVM* [11, 12, 36, 15]. The ones that showed good performance in other fields like language model method and *SVM* also showed high accuracy for authorship classification. For this reason, usually *SVM* has been used to compare the effectiveness of feature sets [12, 15], so in this paper we also use *SVM* for fair comparison between our new feature set and previous feature sets.

Our proposed feature set of *k-ee* can be considered as a variation of tree patterns. In data mining domain, there have been several studies on tree pattern mining [33, 9, 28]. *TreeMiner* [33] is one of the pioneer of mining frequent tree patterns. *CMTreeMiner* [9] mined closed and maximal frequent tree patterns together.

For tree classification, rule-based classifiers (*XRules* [34], *CTC* [38]) and a decision tree based classier (*Tree*^{2} [5]) were proposed. But none of them could be applied to classify sets of trees as documents.

3. PRELIMINARIES

Traditional authorship attribution approaches adopted function words, *POS* tags, and rewrite rules as a feature set to build a classification model. Even though they achieved good accuracy, there still existed room to find a more meaningful feature set to improve the performance. In this section, we describe rewrite rules which are somewhat complex syntactic structures that hold more syntactic information than the other two feature sets. Secondly, we define our new feature set of *k-ee* subtree patterns.

3.1 Rewrite Rule

In [3], rewrite rules were considered to be building blocks of a syntactic tree, just as words are building blocks of a sentence. Here, a *syntactic tree* is a rooted and ordered tree which is labeled with *POS* tags that represents the syntactic structure of a sentence. Its interior nodes are labeled by nonterminals of the grammar, and the leaf nodes are labeled by terminals.

Compared to previous approaches that utilized function words and *POS* tags, rewrite rules can hold functional structure information of the sentence. In linguistics, a rewrite rule is in the form of "$X \rightarrow Y$" where X is a syntactic category label and Y is a sequence of such labels such that X can be replaced by Y in generating the constituent structure of a sentence. For example, "$NP \rightarrow DT{+}JJ{+}JJ{+}NN$" means that a noun phrase (NP) consists of a determiner (DT) followed by two adjectives (JJ) and a noun (NN).

There is a limit when using rewrite rules as features of a classification model. First, because of the restriction that the entire rule cannot be broken into smaller parts, no similarity between rules are considered. A large number of slightly different rules are all counted as independent features. For instance, a rewrite rule "$NP \rightarrow DT{+}JJ{+}NN$", missing one JJ from the above example, becomes a separate rewrite rule. Second, since a rewrite rule is a two-level tree structure, it is not enough to hold most of the syntactic structure information of a sentence. For example, the relationships between rewrite rules are missing, which can hold more refined syntactic information. For these reasons, we developed a new feature set of *k-ee* tree patterns that are flexible and complex enough to represent the syntactic structure information of a sentence.

3.2 k-Embedded-Edge Subtree Pattern

To overcome the drawbacks of the feature sets used in previous approaches, we extended the definition of the rewrite rule to form a new feature set. Based on the analysis of the rewrite rule, a new feature should be a multi-level tree structure to hold the novel information of the syntactic structure of a sentence. Moreover, it should be allowed to contain only a part of a rewrite rule. Induced subtree patterns of a syntactic tree were one of the candidate feature set which satisfied both conditions. But, our pilot experiments showed that a small number of combinations of those induced subtree patterns could achieve even higher accuracy, which motivated us to define *k-ee* subtree patterns for our new feature set as follows.

Definition 1. We define a tree t to be an ***induced subtree*** of a tree s if there exists an identity mapping from t to s preserving all parent-child relationships between the nodes of t. We define an edge e of a tree s to be ***embedded*** *iff* e is a pair of two nodes of s with an ancestor-descendant (not parent-child) relationship. We define a ***k-embedded-edge subtree*** (***k-ee subtree***) t of a tree s to be a set of induced subtrees of s that can be connected by at most k embedded edges.

Since we allow a *k-ee* subtree pattern to be not only a two-level but also a multi-level subtree structure, the number of *k-ee* subtree patterns would be exponential on the number of trees and their sizes. We define a minimum support to ensure we only mine general common patterns that will be applicable to test data thus avoiding overfitting.

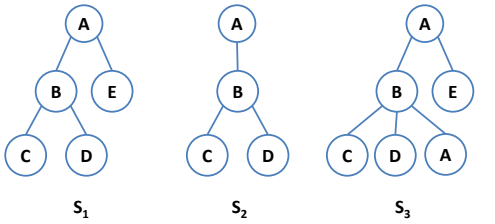


Figure 2: A toy example of a database $\mathcal{D}$ with three syntactic trees

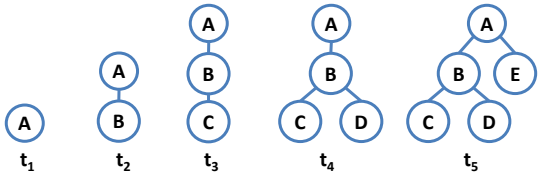


Figure 3: Examples of frequent k-*ee* patterns in $\mathcal{D}$ when $k = 0$ and $\alpha = 2$

Definition 2. We define the ***support*** of a k-*ee* subtree pattern t (denoted by $sup(t)$) to be the total number of syntactic trees of sentences in training data that contains t. We say t is ***frequent*** *iff* $sup(t) \geq \alpha$ for a user-specified minimum support threshold α.

As common words or function words were studied as features for authorship classification in previous works, frequent patterns share the philosophy that more general features are preferred to discriminate the writing styles of the authors.

Figure 2 shows a toy database $\mathcal{D}$ of three syntactic trees. Given minimum support threshold 2, all five 0-*ee* subtree patterns presented in Figure 3 become frequent. For example, patterns t_1, t_2, t_3, and t_4 appears in all three syntactic trees, so their supports are all 3. Pattern t_5 only appears in S_1 and S_3, so its support becomes 2.

Even though we only use frequent k-*ee* subtree patterns as a feature set for a classification model, the potential number of patterns can still become a bottleneck. To address this problem, we introduce the concept of a closed pattern in order to prevent generating redundant patterns; in this way we can summarize frequent patterns into a smaller set of closed patterns without any loss of information.

Definition 3. We define a k-*ee* subtree pattern t to be ***closed*** if there exists no tree pattern t' that contains t with $sup(t') = sup(t)$.

For example, two 0-*ee* subtree patterns t_4 and t_5 in Figure 3 are closed in the toy database $\mathcal{D}$ since their superpatterns have smaller support. Patterns t_1, t_2, and t_3 are not closed since they have the same support with their superpattern t_4.

We will explain how to mine closed k-*ee* subtree patterns efficiently utilizing several pruning techniques in Section 4. In this way, using closed and frequent k-*ee* subtree patterns as a new feature set not only reduces the size of the feature set but also makes our authorship classification framework more scalable.

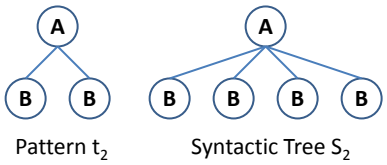


Figure 4: An example of overcounting of overlapped k-*ee* subtree pattern occurrences

3.3 Frequency Measure of k-ee Subtree

The frequency of a pattern within a document (or a set of syntactic trees) is quite important in the sense that it can be a good measure to discriminate the writing styles of different authors. Previously well-known features such as function words, *POS* tags, and rewrite rules adapted *bag-of-words* approach that used the number of their occurrences in a document as their frequency measure. However, the k-*ee* subtree patterns cannot simply adapt the same frequency measure because it generates many overlapped occurrences, which would lead to an exaggerated frequency measure. Overlapped patterns appear because we consider all kinds of subtrees allowing several embedded edges. Figure 4 is an illustration of this overcounting problem. The syntactic tree S has only one A and four Bs, but the number of occurrences of pattern t becomes 6. More generally, if A has n Bs as its children in S, then the occurrence count of pattern t becomes $O(n^2)$. Since we allow k embedded edges for a k-*ee* subtree pattern, this overcounting problem will be even more amplified.

Our observation that a document is parsed into a set of syntactic trees (of sentences) gave us an insight to define the frequency measure of a k-*ee* subtree pattern in a different way by counting the number of syntactic trees of a document that contain the pattern.

Definition 4. We define the ***frequency*** of a k-*ee* subtree pattern t in a document d (denoted by $freq(t, d)$) to be the fraction of the number of syntactic trees of sentences in d that contains t.

For example, if a document d is composed of S_1, S_2, and S_3 in Figure 2, then the frequencies of patterns t_1, t_2, t_3, and t_4 (in Figure 3) in d become all 1 while the frequency of a pattern t_5 in d becomes 2/3.

Note that our feature value is a normalized score in the sense that we only consider the fraction of the number of sentences in a document. In this way, we can remove the effect of different document lengths.

3.4 k-ee Subtree-based Authorship Classification

We propose a k-*ee*-subtree pattern-based authorship classification framework with the following four steps: (1) Convert each document into a set of syntactic trees. As mentioned earlier, several high-quality parsing tools have been developed recently. (2) Mine frequent k-*ee* subtree patterns of the syntactic trees from the training data. There are several reasons we use only frequent patterns. First, we do not assume the parser works perfectly with no error, but we do assume it works with a reasonable accuracy. A small rate of error might produce strange patterns with low support. Therefore, if we only use frequent patterns, we can reduce

the influence of parsing errors. Second, using patterns with low support as features may cause overfitting and subsequently harm the classification accuracy. Statistically, using frequent patterns of training data as features for the classification model generalizes well to the test data, since frequent patterns of training data have a higher chance to also appear in test data. (3) Select discriminative patterns from the frequent k-*ee* subtree patterns. Depending on the user specified minimum support threshold, we might get a large number of frequent patterns which may again cause overfitting. Therefore, we carefully choose only a small number of non-redundant and highly discriminative patterns as the features for the classification model. (4) Construct the classification model with the discriminative patterns and training data.

4. K-EE SUBTREE PATTERN MINING

In the previous section, we explained the reasons to use k-*ee* subtree patterns as a new feature set of authorship classification. These patterns hold more profound syntactic information (than other features including rewrite rules) and are flexible enough to consider partial matching of the syntactic trees. Even though the k-*ee* subtree patterns are confined to be frequent and closed, the number of patterns can still be very large. Therefore, the next task is to mine these patterns efficiently.

In this section, we introduce a k-*ee* subtree pattern mining method that (i) finds the frequent and closed patterns efficiently and (ii) captures their frequencies in each document. We do not generate candidate k-*ee* subtree patterns and check for frequent and closed attributes. Instead, we find a frequent k-*ee* subtree pattern, and extend it by adding a node (that is guaranteed to be frequent) in a depth-first search manner. Depth-first search pattern expansion enables several pruning techniques for closed and frequent pattern mining. We first introduce how to efficiently find a frequent node for pattern extension, and then explain the pruning techniques for closed pattern mining.

4.1 Pattern-Growth Approach

Previous apriori-based approaches for pattern mining generated a huge number of patterns and re-scanned the entire database each time the size of candidate patterns were increased to verify whether they were frequent. Recently, several studies were conducted on the pattern-growth approach by using the projected database in sequence and tree pattern mining [24, 39]. In this study, we adapt these pattern-growth techniques for frequent k-*ee* subtree pattern mining.

Instead of the apriori-based expensive candidate generation and test framework, we follow the following steps for pattern-growth approach. First, find a size 1 frequent k-*ee* subtree t in the training dataset $\mathcal{D}$. Second, project the postfix of each occurrence of t in the syntactic trees of $\mathcal{D}$ into a new database $\mathcal{D}_t$. Here, we say a ***postfix*** of an occurrence of t in a syntactic tree s to be the forest of the nodes of s appearing after the occurrence of t in a pre-order scan of s. Third, find a frequent node v in $\mathcal{D}_t$ that can be attached to the rightmost path of t that forms a k-*ee* subtree pattern. Once v is frequent in $\mathcal{D}_t$, it ensures that the extended pattern is also frequent, so we do not need to scan the whole database $\mathcal{D}$ again. The reason we only search for a frequent node that can be attached to the rightmost path of t is to avoid generating duplicated patterns in the mining process. Note that, in this study, we consider a node v attached to t by an induced edge forms a different pattern from the one attached by an embedded edge because of the k-embedded edge restriction. So we consider each case separately. Fourth, recursively go back to second step with the extended pattern for every frequent node we found. Fifth, recursively go back to the second step to expand all the other size-1 frequent k-*ee* subtrees.

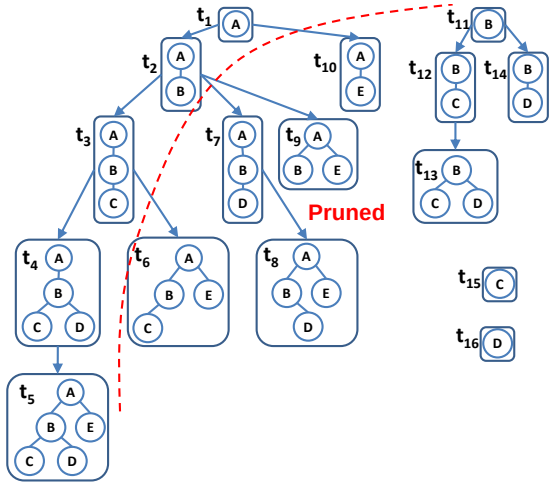


Figure 5: Pattern growth of k-*ee* subtree patterns using pruning with minimum support 2 when k=0

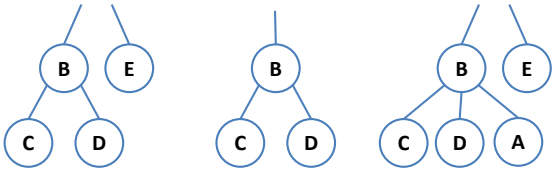


Figure 6: Projected database of t_1

4.2 Pruning Methods

Figure 5 shows an example of the pattern-growth approach to mine 0-*ee* subtree patterns of a database of three syntactic trees described in Figure 2 when the minimum support threshold is 2. Each pattern is indexed in pattern-generation order. We first search for size-1 frequent patterns, which are t_1, t_{11}, t_{15}, and t_{16} in this case. We choose t_1 as a starting point, and find frequent nodes that can be attached to t_1 from its projected database described in Figure 6. We find nodes B and E are frequent, and we extend t_1 to t_2 by adding a node B. Once all frequent 0-*ee* subtree patterns that extends t_2 are all mined, then we extend t_1 to t_{10} by adding a node E. Similar procedures are recursively performed until we mine all frequent patterns.

In our pattern-growth approach, the projected database of a pattern t keeps shrinking as the mining process moves on and t becomes a bigger superpattern. Note that we do not physically create projected databases. In fact, instead of keeping physical copy of postfix data, we use a *pseudo-projection* that only stores a pointer to the syntactic tree and the offset of each node of a pattern occurrence in the syntactic tree to save memory and make the procedure more efficient.

Algorithm 1: Procedure *ClosedMine* to mine *k*-ee closed subtree patterns

input : Tree data set $\mathcal{D}$, minimum support θ
output: Closed k-ee subtree patterns $\mathcal{C}$

```
1 foreach frequent vertex t ∈ D do
2 |  ClosedMine_Sub(t,D_t,θ);
3 end
```

Algorithm 2: Subprocedure ClosedMine_Sub used for *ClosedMine*

```
1  if t satisfies BEP condition then return;
2  if no t' ∈ B_t sentence-matches with t then
3  |  C ← C ∪ {t};
4  end
5  foreach t' ∈ B_{t_right} do     /* bottom up to enable FEP
   pruning */
6  |  if t' satisfies FEP condition then break;
7  |  if sup(t') ≥ θ then
8  |  |  ClosedMine_Sub(t',D_{t'},θ);
9  |  end
10 end
```

After we perform the pattern-growth method to mine all frequent *k-ee* subtree patterns, we can remove the patterns which are not closed. Instead of the inefficient two-step approach, we can integrate several well-known pruning techniques for semi-structured data mining [32, 29, 9] into the pattern-growth method to output only closed patterns. The common intuition of the pruning methods is that we only need to check immediate supertrees of a tree pattern t not the whole supertree for the closure checking. In this paper, we adapt the *blanket* convention of [9] to describe pruning techniques for closed *k-ee* subtree pattern mining as follows.

Definition 5. Define the ***blanket*** of a *k-ee* subtree pattern t (denoted by B_t) by the set of supertrees of t that has one more node than t. For a pattern $t' \in B_t$, we denote $t' \backslash t$ to be an additional node v of t' that is not in t. Here, $t' \backslash t$ represents not only the vertex label of v, but also its position and the type of edge connection (either induced or embedded) between t and v. We define the ***right-blanket*** of t (denoted by B_t^r) as a subset of B_t where $t' \in B_{t\_right}$ *iff* $t' \backslash t$ is the rightmost vertex of t'. We define the ***left-blanket*** of t (denoted by B_t^l) by $B_t^l = B_t - B_t^r$. For $t' \in B_t$, we define t' and t to be ***occurrence-matched*** if, for each occurrence of t in a database, there is at least one corresponding occurrence of t'. We define t' and t to be ***sentence-matched*** if for any syntactic tree s of a sentence in $\mathcal{D}$ that contains t it also contains t'.

For example, in Figure 5, pattern t_4 is in the blanket of t_7 since t_4 is a superpattern of t_7 by one more node C. And also, pattern t_4 is in both $B_{t_7}^l$ and $B_{t_3}^r$. Since $t_4 \in B_{t_7}$ and each occurrence of t_7 is contained in an occurrence of t_4, we say t_4 and t_7 are occurrence-matched.

The following two pruning techniques are based on occurrence-level matching. *Backward Extension Pruning*(BEP) checks the occurrence matching of the current mining tree pattern t with previously mined supertrees of t, and *Forward Extension Pruning*(FEP) checks the occurrence matching of t with the supertrees of t that will be mined later.

PROPOSITION 1. **(Backward Extension Pruning)** *For a k-ee subtree pattern t, if there exists a supertree $t' \in B_t^l$ such that t and t' are occurrence-matched, then neither t nor any supertrees of t as extensions of any node of its rightmost path can be closed.*

For example, pattern t_7 and its descendants in Figure 5 are pruned since t_4 is in the left blanket of t_7, and t_4 and t_7 are occurrence-matched which satisfies the *BEP* condition. Similarly, t_{10}, t_{11}, t_{15}, t_{16} and their descendants are pruned because of *BEP* criterion.

PROPOSITION 2. **(Forward Extension Pruning)** *For a k-ee subtree pattern t, if there exists a supertree $t' \in B_t^r$ such that t and t' are occurrence-matched and the parent of $t' \backslash t$ is v (where v is a vertex on the rightmost path of t), then neither t nor any supertrees of t as extensions of any proper ancestor node of v can be closed.*

For example, pattern t_6 and its descendants in Figure 5 are pruned since all conditions for FEP are satisfied as follows: (i) t_4 is in the right-blanket of t_3 (ii) D is $t_4 \backslash t_3$ (iii) A is the proper ancestor node of D's parent node in t_4 (iv) t_6 is an extension of t_3 by adding a node E at A. Similarly, pattern t_9 and their descendants are pruned because of the *FEP* criterion.

BEP described in Proposition 1 means that once we find a pattern t' is in the left-blanket of t that occurrence-matches with t, then we do not have to perform pattern-growth of t, because a pattern extension in the pattern-growth approach is performed in a depth-first traversal manner. *FEP* described in Proposition 2 is a simple corollary of the *BEP*.

Algorithm 1 and 2 describe how to incorporate the pruning methods *BEP* and *FEP* into pattern-growth approach to mine closed and frequent *k-ee* subtree patterns. In Algorithm 2, line 5 and 8 ensures the algorithm to work in a pattern-growth way. We check *BEP* condition at line 1, and *FEP* condition at line 6. In this way, we do not have to generate all frequent *k-ee* subtree patterns to mine closed patterns.

5. DISCRIMINATIVE K-EE SUBTREE PATTERN SELECTION

In Section 3, we developed an algorithm to mine closed and frequent *k-ee* subtree pattern, but there may still be too many resulting patterns. In this section, we present how to carefully select discriminative patterns from among the closed and frequent patterns in order to reduce the size of the feature set and to improve the performance of the classifier.

Based on the study that the patterns with high *Fisher* score can help improving the classification performance [7], we use it in our study to evaluate the discriminative power of a *k-ee* subtree pattern. The *Fisher* score is defined as

$$Fr = \frac{\sum_{i=1}^{c} n_i(\mu_i - \mu)^2}{\sum_{i=1}^{c} n_i \sigma_i^2}$$

where n_i is the number of data samples in class i, μ_i is the average pattern frequency in class i, σ_i is the standard deviation of the pattern frequency in class i, and μ is the

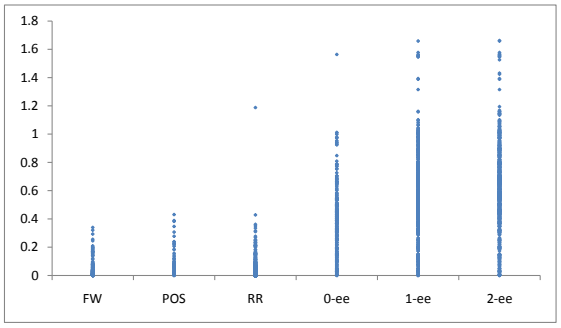


Figure 7: Fisher score distribution of various feature sets

Algorithm 3: Procedure *WSMine* to mine discriminative k-ee subtree patterns

input : Tree data set $\mathcal{D}$, min_sup θ
output: Discriminative k-ee features F of $\mathcal{D}$

```
C ← ClosedMine(D,θ);
while (C ≠ ∅) or (D ≠ ∅) do
    Select top-1 discriminative pattern t from C;
    F ← F ∪ {t};
    D ← D − {trees that are covered by δ features in F};
    C ← C − {t};
end
```

average pattern frequency in the whole dataset. A pattern will have a large *Fisher* score if it has similar values within the documents of the same class and very different values across the documents of different classes, at the same time.

Figure 7 presents *Fisher* score distributions of various feature sets such as function words (FW), POS tags (POS), rewrite rules (RR), and k-*ee* subtree patterns for k=0, 1, and 2 (0-ee, 1-ee, and 2-ee, respectively). We can easily see that the highest scores are mostly from k-*ee* subtree patterns, which implies that they can be more meaningful than other features. In fact, in the experiments, our k-*ee* subtree patterns achieved highest accuracy for all datasets.

Based on this *Fisher* score measure, we perform the feature selection procedure in a sequential coverage way as follows. We describe this procedure in Algorithm 3. We select the top scored pattern which covers at least one syntactic tree of the dataset and remove it from the list of the patterns. Moreover, any syntactic tree that is covered by at least δ features will be removed from the dataset. Here, *delta* is a feature coverage threshold introduced in [7]. It allows multiple patterns to represent a tree, which is known to improve the classification accuracy. Third, we go back to the second step until either the dataset becomes empty or no more patterns are left.

Once the feature selection procedure is complete, we get a small number of discriminative, closed, and frequent k-*ee* subtree patterns. Considering these patterns as a feature set, we express a document as a vector representation assigning a feature value by the frequency of the pattern described in Definition 4, and learn a classification model.

6. EXPERIMENTS

In this section, we present empirical evaluation results to validate the performance of our authorship classification framework. In particular, we conduct experiments on news articles and movie reviews. The experiments are designed to test whether our k-*ee* subtree patterns, as a new feature set, are useful for authorship classification.

Table 1: Statistics of News Articles

	# of documents	# of sentences	# of words
N1	100	3710	77501
N2	100	2666	59745
N3	100	5587	114337
N4	100	7198	129867
Total	400	19161	381450

Table 2: Statistics of Movie Reviews

	# of documents	# of sentences	# of words
M1	578	16508	423749
M2	567	15108	414295
M3	597	15320	357301
M4	415	4150	104337
Total	2177	51086	1299682

6.1 Datasets

We collected two different kinds of documents from a public data collection *The New York Times*: news articles and movie reviews. We got four authors with 400 documents for news articles, and four authors with around 2,000 documents for movie reviews.

For the news articles, we chose two journalists Eric Dash (N1) and Jack Healy (N2) from business department, and two other journalists Denise Grady (N3) and Gina Kolata (N4) from health department, who were one of the main contributors in their departments. The reason we collected documents in this way is because the journalists in the same department are likely to write articles in the same topic and genre using similar words. The statistics of each journalist are shown in Table 1.

For the movie reviews, we chose four main movie critics of *The Times*: A. O. Scott (M1), Manohla Dargis (M2), and Stephen Holden (M3), and Jeannette Catsoulis (M4). The reason we collected this data is because movie reviews of the same movie are likely to be in the same topic and genre using similar words. The statistics of each critics are shown in Table 2.

6.2 Evaluation Methodology

To evaluate the performance, we paired the authors of each domain and conducted binary classification on these 12 different datasets. For each dataset, we conducted 5-fold cross validation, and averaged the accuracy as a measure of the performance. For each fold, training data was used to mine the syntactic features and to get a classification model while test data was only used for prediction purpose. In this way, our evaluation ensured that there is no information leak from the test data for the classification task.

To show how effectively our new feature set works, we compared the authorship classification performance with other syntactic features such as function words, *POS* tags, and rewrite rules. As for function words, we took the list of 308 function words from [21]. We used 70 *POS* tags generated

Table 3: Number of Features

Domain	RR	0-ee	1-ee	2-ee
News Articles	3929	280.83	560.23	789.93
Movie Reviews	9029.2	557.87	1348.9	2074.5

from the stanford parser [18]. The number of features of the other feature sets are presented in Table 3. For each feature set and for each dataset, we computed the average value of the number of distinct features of 5-fold training data. In the table, we showed the average of the number of distinct features for each domain. The difference of the number of features between different domains implies that movie reviews are written in more sophisticated way than news articles. That also implies indirectly that it would be harder to classify movie reviews than news articles. We see that rewrite rules are using the biggest number of features.

We used the occurrences of each feature as a feature value for the syntactic features except k-ee subtree patterns which used a new frequency measure defined in Definition 4. For the fair comparison, we used the same classifier, linear-kernel *SVM* (with the parameter tuned for the best performance of each feature set), which was previously shown to work reliably with high accuracy on authorship classification [11].

6.3 Performance Evaluation

Authorship classification accuracies for various feature sets are presented in Table 4 and 5. All experimental results of k-ee subtree pattern-based classification used (relative) minimum support threshold 0.1 for frequent pattern mining and sequential coverage threshold 10 for discriminative pattern mining by default. We can easily find that our proposed feature set of k-ee subtree patterns, especially for $k = 1, 2$, achieved the highest accuracies for most of the datasets. We see that using embedded edges can help to enhance the authorship classification performance, but we cannot say more embedded edges would get better performance. For a higher number of embedded edges (k), even we utilize several pruning techniques, it is intractable to mine them all. Moreover, higher k sometimes tends to overfit to training data that might degrade the accuracy performance. We can conclude that a small number of embedded edges is enough to achieve high performance of classification task for both in accuracy and efficiency aspects.

For both data collections of news articles and movie reviews, all feature sets showed similar tendencies. 1-ee and 2-ee showed the highest accuracies of news article datasets and movie review data collections repectively, while POS got the worst accuracies for both data collections. Among 12 datasets of experiments, N12, N34 and M12 showed bad performances for all feature sets. Analyzing statistics of data collections in Table 1 and 2, we see that the classes in them has similar number of words which indirectly shows those classes are hard to classify. Especially for dataset N34, both classes of N3 and N4 are from health department of news domain and they have quite a few quotations with informal style of writings which made it the hardest dataset to be classified. It is noticeable that even for this hard dataset, our feature set got the highest accuracy with a big gap of performance to the other feature sets.

We calculated the standard deviation of the accuracies to show how reliable the feature sets are, and found that k-ee subtree patterns achieved consistent results for both data collections.

Overall, we conclude that k-ee subtree patterns are meaningful features for authorship classification which works reliably for real life data collections and achieves high accuracy.

Table 4: Accuracy Comparisons (News Articles)

	FW	POS	RR	0-ee	1-ee	2-ee
N12	91.5	87	94	**96**	95	95.5
N13	94	85	91	97.5	**98**	97.5
N14	95.5	92.5	96	94.5	**96.5**	95
N23	95	92.5	92.5	96.5	98.5	**99**
N24	97	95.5	97.5	**98.5**	**98.5**	**98.5**
N34	80.5	67.5	67.5	88.5	**90**	**90**
AVG	92.3	86.7	89.8	95.3	**96.1**	96.0
STD	6.04	10.16	11.15	3.57	**3.28**	3.31

Table 5: Accuracy Comparisons (Movie Reviews)

	FW	POS	RR	0-ee	1-ee	2-ee
M12	92.8	81.0	88.0	92.48	**94.26**	94.22
M13	93.6	92.5	92.7	95.22	95.06	**95.8**
M14	92.1	88.0	94.2	97	97.4	**97.7**
M23	94.4	92.8	94.8	97.58	**97.92**	97.58
M24	93.1	91.0	92.9	95.22	96.04	**96.32**
M34	93.1	88.6	94.9	97.12	**97.22**	97.12
AVG	93.2	89.0	92.9	95.8	96.3	**96.5**
STD	*0.77*	4.40	2.59	1.90	1.45	1.32

7. CONCLUSION

In this paper, we proposed a novel solution for an authorship classification problem by mining discriminative closed *k-ee* subtree patterns. First, we designed a new feature set of *k-ee* subtree patterns which contains more meaningful syntactic structures of a sentence than previous feature sets which are based on simple forms of syntactic features including function words, *POS* tags, and rewrite rules. To mine *k-ee* subtree patterns, we developed a closed frequent *k-ee* tree mining algorithm by use of several pruning techniques. We performed a *Fisher* score based feature selection procedure on top of those mined patterns. This small set of discriminative patterns could effectively classify the documents based on their authorship.

Experimental study has been performed on two real datasets, news articles and movie reviews, from *The New York Times* public data corpus. These data collections were carefully chosen to ensure to be in the same genres using similar terms. Our *k-ee* subtree pattern based classification achieved the best results compared to other feature sets such as function words, *POS* tags, and rewrite rules.

In future research, we want to develop a way to directly mine discriminative *k-ee* subtree patterns, not generating all closed patterns. Usually, discrimiative patterns selected from closed patterns with low minimum support threshold θ show better accuracy, but it is hard to find an optimized θ since the mining cost increases exponentially when θ becomes lower. A directive way of mining discriminative pattern might work without specifying θ which would guarantee high quality of discriminative patterns.

8. REFERENCES

[1] S. Argamon and S. Levitan. Measuring the usefulness of function words for authorship attribution. In *ACH/ALLC*, 2005.

[2] S. Argamon, M. Šarić, and S. S. Stein. Style mining of electronic messages for multiple authorship discrimination: first results. In *KDD*, 2003.

[3] H. Baayen, H. van Halteren, and F. Tweedie. Outside the cave of shadows: using syntactic annotation to enhance authorship attribution. *Literary and Linguist Computing*, 11(3):121–132, 1996.

[4] S. Bloehdorn and A. Moschitti. Structure and semantics for expressive text kernels. In *CIKM*, 2007.

[5] B. Bringmann and A. Zimmermann. Tree^2 - decision trees for tree structured data. In *PKDD*, 2005.

[6] S. Burrows, A. L. Uitdenbogerd, and A. Turpin. Application of information retrieval techniques for source code authorship attribution. In *DASFAA*, 2009.

[7] H. Cheng, X. Yan, J. Han, and C.-W. Hsu. Discriminative frequent pattern analysis for effective classification. In *ICDE*, 2007.

[8] H. Cheng, X. Yan, J. Han, and P. S. Yu. Direct discriminative pattern mining for effective classification. In *ICDE*, 2008.

[9] Y. Chi, Y. Xia, Y. Yang, and R. R. Muntz. Mining closed and maximal frequent subtrees from databases of labeled rooted trees. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):190–202, 2005.

[10] O. de Vel, A. Anderson, M. Corney, and G. Mohay. Mining e-mail content for author identification forensics. *SIGMOD Record*, 30(4):55–64, 2001.

[11] J. Diederich, J. Kindermann, E. Leopold, and G. Paass. Authorship attribution with support vector machines. *Applied Intelligence*, 19(1-2):109–123, 2003.

[12] M. Gamon. Linguistic correlates of style: authorship classification with deep linguistic analysis features. In *COLING*, 2004.

[13] A. M. García and J. C. Martín. Function words in authorship attribution studies. *Literary and Linguistic Computing*, 22(1):49–66, 2007.

[14] J. Grieve. Quantitative authorship attribution: An evaluation of techniques. *Literary and Linguistic Computing*, 22(3):251–270, 2007.

[15] G. Hirst and O. Feiguina. Bigrams of syntactic labels for authorship discrimination of short texts. *Literary and Linguistic Computing*, 22(4):405–417, 2007.

[16] D. L. Hoover. Statistical stylistics and authorship attribution: an empirical investigation. *Literary and Linguistic Computing*, 16(4):421–444, 2001.

[17] D. L. Hoover. Another perpective on vocabulary richness. *Computers and the Humanities*, 37(2):151–178, 2003.

[18] D. Klein and C. D. Manning. *The Standford parser: A Statistical Parser*, 2002. `http://nlp.stanford.edu/software/lex-parser.shtml`.

[19] D. Lo, H. Cheng, J. Han, S.-C. Khoo, and C. Sun. Classification of software behaviors for failure detection: a discriminative pattern mining approach. In *KDD*, 2009.

[20] T. C. Mendenhall. Spelling checkers,spelling correctors and the misspellings of poor spellers. *Science*, 11(214):237–246, 1887.

[21] R. Mitton. Spelling checkers,spelling correctors and the misspellings of poor spellers. *Information Processing and Management*, 23(5):495–505, 1987.

[22] A. Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*, 2006.

[23] F. Mosteller and D. L. Wallace. *Inference & Disputed Authorship: The Federalist*. Addison Wesley, 1964.

[24] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, U. Dayal, and M. chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *ICDE*, 2001.

[25] J. Rudman. The state of authorship attribution studies: Some problems and solutions. *Computers and the Humanities*, 31(4):351–365, 1998.

[26] C. Sanderson and S. Guenter. Short text authorship attribution via sequence kernels, markov chains and author unmasking: an investigation. In *EMNLP*, 2006.

[27] E. Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556, 2009.

[28] A. Termier, M.-C. Rousset, M. Sebag, K. Ohara, T. Washio, and H. Motoda. Dryadeparent, an efficient and robust closed attribute tree mining algorithm. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 20(3):300–320, 2008.

[29] J. Wang and J. Han. Bide: Efficient mining of frequent closed sequences. In *ICDE*, 2004.

[30] K. Wang, Z. Ming, and T.-S. Chua. A syntactic tree matching approach to finding similar questions in community-based qa services. In *SIGIR*, 2009.

[31] X. Yan, H. Cheng, J. Han, and P. S. Yu. Mining significant graph patterns by leap search. In *SIGMOD*, 2008.

[32] X. Yan and J. Han. Closegraph: mining closed frequent graph patterns. In *KDD*, 2003.

[33] M. J. Zaki. Efficiently mining frequent trees in a forest. In *KDD*, 2002.

[34] M. J. Zaki and C. C. Aggarwal. Xrules: an effective structural classifier for xml data. In *KDD*, 2003.

[35] Y. Zhao and J. Zobel. Effective and scalable authorship attribution using function words. In *AIRS*, 2005.

[36] Y. Zhao, J. Zobel, and P. Vines. Using relative entropy for authorship attribution. In *AIRS*, pages 92–105, 2006.

[37] R. Zheng, J. Li, H. Chen, and Z. Huang. A framework for authorship identification of online messages: Writing-style features and classification techniques. *Journal of the American Society for Information Science and Technology*, 57(3):378–393, 2006.

[38] A. Zimmermann and B. Bringmann. Ctc — correlating tree patterns for classification. In *ICDM*, 2005.

[39] L. Zou, Y. Lu, H. Zhang, R. Hu, and C. Zhou. Mining frequent induced subtrees by prefix-tree-projected pattern growth. In *WAIMW*, 2006.

Pattern Selection Problems in Multivariate Time-Series using Equation Discovery

Arne Koopman, Arno Knobbe, Marvin Meeng
LIACS
Universiteit Leiden
akoopman@liacs.nl

ABSTRACT

In this paper, we present a method for pattern selection in collections of patterns discovered in multivariate time-series. Because our data is continuous in nature, the pattern language we consider is somewhat out of the ordinary, compared to the common discrete patterns considered in the data mining field. An equation discovery system is employed to generate either regular algebraic equations, or more complex differential equations. As the equation discovery system generates a collection of equations per target variable, and we require equations for each variable, we are dealing with an abundance of equations, quite likely with serious levels of redundancy. The method presented here selects a subset of equations by considering to what extent the different variables are covered by the selected equations, while optimising the relevance of variables within the equations. As such, the equation selection method returns a concise set of equations, that captures the dependencies between the different time-series well, while minimizing redundancy. The work in this paper is inspired by the new InfraWatch project, which deals with high-resolution sensor data from a highway bridge. The 145 sensors (sensing structural characteristics such as stretch, vibration and temperature) are distributed fairly densely over the bridge, such that adjacent sensors are likely to show correlated signals. Especially in an exploratory setting, one would be interested in a small collection of prototype sensors with associated equations for how these prototypes are related to other sensors in the vicinity. In the experimental section, we demonstrate how the sensors can be modeled by (differential) equations, and how the equation selection method picks relevant equations that models structural properties of the bridge sensibly.

1. INTRODUCTION

This paper is concerned with multivariate time-series, specifically with data collected by a series of sensors measuring at a steady frequency. In the typical case, these sensors measure the state of a certain system at various locations, such that the measured signals will show a certain degree of correlation. Our aim is to model such correlations by means of patterns discovery. Compared to traditional pattern discovery, our data is challenging in two particular ways. First, the data is of *continuous* nature, which excludes the majority of (discrete) pattern discovery approaches. Second, the data is measured as a function of *time*, which implies a certain dependency between consecutive measurements for a given sensor. In order to deal with these two challenges, we employ an equation discovery system that is capable of finding both regular algebraic equations (the continuous aspect) as well as differential equations (the temporal aspect). The outcome of the equation discovery process is a (potentially large) collection of equations which model the value of one sensor as a function of a small number of other sensors. Such equations, and how well they approximate reality, can be used by analysts to find elementary relations between components of the observed system, and may also suggest possible redundancies in the sensor network.

The equation discovery system in question is the *Lagramge* system, developed as an extension of the earlier Lagrange system by Todorovski and Džeroski [4, 5, 15]. The system considers a grammar of well-formed equations, and constructs candidate equations in a fashion reminiscent of ILP or multi-relational systems [8, 5]. That is, the essentially structural descriptions of the equations are constructed top-down, with progressively more complex equations being built by adding new variables (the sensors) and functions. A specific difference to said relational approaches is the component of Lagramge which fits parameters of the equations by means of the downhill simplex or Levenberg-Marquart algorithms [13]. As such, the system combines a pattern discovery-inspired component with an error-minimization component. The typical setting of a Lagramge run is to select a target sensor s_x and specify a declarative bias (the equation grammar) after which the system returns a list of possible equations. Each equation includes a right-hand side which involves a number of sensors. The equations differ in the sensors involved, and in the constants that were fitted, and consequently in the error on the data.

Although Lagramge produces the desired information (a collection of equations modeling local dependencies in the observed system), it suffers from a problem that is common to most pattern discovery system: an abundance of results. This abundance comes from a number of sources. First of all, different equations may model the dependencies equally well, approximately. This redundancy is especially apparent if two or more sensors are highly correlated, such

UP'10, July 25th, 2010, Washington, DC, USA.

Figure 1: (left) Aerial picture of the situation of the Hollandse Brug, which connects the 'island' Flevoland to the province Noord-Holland, and the adjacent railway bridge. (right) Some of the sensors attached to the underside of the bridge.

that one of the sensors can simply be exchanged for another, without essentially affecting the model. Furthermore, there may be terms in the equation that do not substantially contribute to the overall fit, for example by having insignificant constants. Finally, there is redundancy produced by running Lagramge once for each potential target sensor (all sensors once, if need be), such that one sensor may be modeled in terms of another, and vice versa. Note that all of these instances of pattern-redundancy appear in other pattern discovery settings as well, including overlapping conditions, irrelevant conditions and so on. Observing this analogy between the equation selection problem and the pattern selection problem, we propose to select a small but relevant collection of equations, using a method that is inspired by a number of recent pattern selection methods [9, 1].

The patterns selection methods mentioned are centered around the idea of greedy forward selection of the patterns. Starting with an empty set of patterns C, all remaining patterns are considered in turn, and a pattern f is added if it improves the quality of $C \cup f$ substantially. This process continues until either some stopping-criterion is met, or all patterns have been included. In the case of patterns (which are typically interpreted as binary features), the *quality* of $C \cup f$ is often a measure of the joint entropy of $C \cup f$ [9], or the mutual information between C and f [1]. As an alternative, more syntax-oriented interpretations of quality may be employed, for example to optimize coverage of all items in itemsets. This is one of the approaches that we will assume in our equation selection method. Replacing patterns by equations, we will assume that a specific equation selected in C accounts for all sensors that appear in, both the left and right hand side of, the equation. Therefore, a new equation that features only sensors that do not yet appear in any of the elements of C is a desirable addition to C. Our method thus adds equations that cover as many new sensors as possible.

Our work on equation discovery in multivariate time-series is inspired by a recently started project, called InfraWatch [7]. In this project, we deal with 145 sensors attached to, or embedded in, the concrete of a large highway bridge in the Netherlands (see Section 2). These sensors measure the weather and traffic load on various locations of the bridge at a frequency of 100Hz. Because the sensors are distributed over the bridge, and vibrations are conducted through the rigid structural elements, nearby sensors will be correlated. Furthermore, sensors of different nature (e.g., stretch vs. vibration) may be co-located, such that potentially differential equations may be required to model the difference in physical properties these types of sensors measure. Especially for exploratory purposes, an analysts would be served be a concise, yet informative set of sensors. Although our main application in this paper is related to the InfraWatch project, one could apply the same techniques to other data of similar kind. One example can be found in the Adaptive System Management problem [10, 11], where large collection of *monitors* are continuously measuring the state and health of different components in an IT-system, and clear numeric dependencies, even of differential nature, between the load in certain components exist.

The work presented in this paper is related to vector autoregression (VAR) [6]. In VAR, the goal is to find one model that best describes how multiple time series are related. In contrast, our method focusses on finding multiple simple models that describe how time series are related. Moreover, we aim to find a non-overlapping set of models that do a good job a describing how they are related.

2. INFRAWATCH

The InfraWatch project is centred around an important Dutch highway bridge: the Hollandse Brug. This bridge is located between the Flevoland and Noord-Holland provinces, at the place where the Gooimeer joins the IJmeer (see Figure 1 on the left). It was opened in June 1969, and in April 2007 it was announced that measurements would have shown that the bridge did not meet the quality and security requirements. Repairs were launched in August 2007 and a consortium of companies has installed a monitoring config-

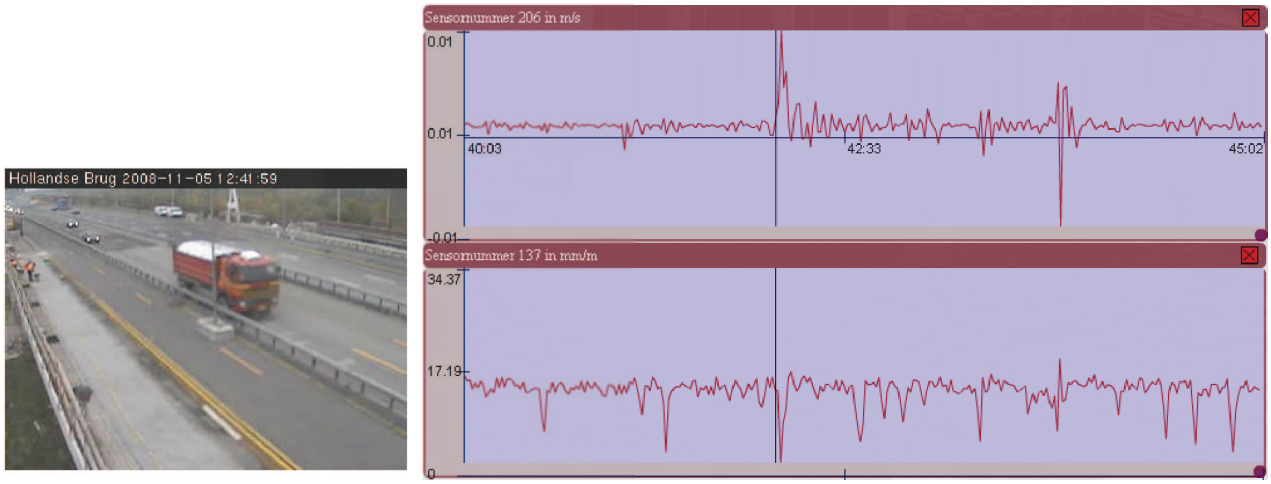


Figure 2: Example of a truck passing the single camera located on the bridge. The graphs show the signal of two sensors, with a vertical bar indicating the time that corresponds to the shown video frame.

uration underneath the Hollandse Brug with the main aim to collect data for evaluating how the bridge responds. The sensor network is part of the strengthening project which was necessary to upgrade the bridge's capacity by overlaying.

The monitoring system comprises 145 sensors that measure different aspects of the condition of the bridge, at several locations along the bridge (see Figure 1 on the right). The following types of sensors are employed:

- 'geo-phones' (vibration sensors) that measure the vertical movement of the bottom of the road-deck as well as the supporting columns.
- strain-gauges embedded in the concrete and attached to the outside, measuring horizontal stress in two directions.
- thermometers embedded in the concrete and attached to the outside.

Furthermore, there is a weather station, and a video-camera that provides a continuous video stream of the actual traffic on the bridge. Additionally, there are plans to monitor the adjacent railway bridge.

Prior to the start of the InfraWatch project, an initial monitoring application was developed by a team of students, that allows the visual inspection of both video and sensor information. The application allows the user to navigate through a selected time-frame, and display the traffic passing over the bridge, while the data over one or more sensors is displayed in synchronised fashion (see Figure 2). The user can select the nature of the sensor as well as the location of it, which does not necessarily have to correspond with the location of the camera. Using this application, it is fairly easy to already observe some patterns in the data. For example, the vertical load data nicely corresponds with heavy vehicles passing.

3. EQUATION SET SELECTION

As part of our first efforts on this data, we aim to find characteristic sensors within the whole sensor sytem S that can be regarded as a representation of a set of other sensors. For example, if sensor s_1 demonstrates the same behaviour as sensors s_2 and s_3, it suffices to consider only s_1 as a prototype for these three sensors.

In order to determine the behaviour of the sensors, we consider the measurement data that is gathered from each sensor. That is, for each sensor s, we have a signal consisting of a stream of continous data that represents some measured aspect of the bridge. We simply denote the signal value of sensor i at timestamp t by: $s_i(t)$. Within the measurement period T, we know at every timestamp $t \in T$ the $s_i(t)$ for each sensor.

Based on these values, we can now find equations that predict the signal values of one sensor based on the reading of others. In theory, we can apply any class of equation that described relations between the sensors. However, as our interest is in finding simple relations between sensors, we initially focus on linear equations. In other words, an equation f_x that approximates s_x over time has the following form:

$$f_x(t) = c_0 + \sum_{s_y \in S} c_y \cdot s_y(t)$$

where $s_y \in S, x \neq y, c_y \in \mathbb{R}$.

We denote the length of an equation f, $L(f)$, as the number of $s_y \in S$, for which $c_y \neq 0$.

Additionaly, we are not so much interested in *any* equation that describes relationships between sensors, but rather in *good* equations. Therefore, we restrict the set of equations to those equations that are able to closely approximate the signal value of the target sensor:

$$\sum_{t \in T} |s_x(t) - f_x(t)| \leq \epsilon \cdot |T|$$

where ϵ is the error threshold. The term $|T|$ compensates for differences in the size of the time window considered, and thus allows a definition of ϵ independent of $|T|$. Given the set of signals, we can now find a set of candidate equations C that match these requirements, by means of Lagramge.

This typically results in many equations, and even worse, many of these equations describe the same behaviour for

Algorithm 1 ForwardSelection

FowardSelection(C)
1. $F = \varnothing; Q = 0$
2. **for all** $f \in\downarrow C$ **do**
3. $\quad F' = F \cup f$
4. $\quad Q' = 0$
5. $\quad$ **for all** $f' \in F'$ **do**
6. $\quad\quad Q' = Q' + simplicity(f)$
7. $\quad$ **end for**
8. $\quad$ **if** $Q' > Q$ and $overlap(F') = 0$ **then**
9. $\quad\quad F = F'; Q = Q'$
10. $\quad$ **end if**
11. **end for**
12. **return** F

Algorithm 2 ForwardSelectionWithPruning

FowardSelectionWithPruning(C)
1. $F = \varnothing; Q = 0$
2. **for all** $f \in\downarrow C$ **do**
3. $\quad F' = F \cup f; Q' = 0$
4. $\quad$ **for all** $f' \in F'$ **do**
5. $\quad\quad Q' = Q' + simplicity(f)$
6. $\quad$ **end for**
7. $\quad$ **if** $Q' > Q$ and $overlap(F') = 0$ **then**
8. $\quad\quad F = F'; Q = Q'$
9. $\quad$ **else**
10. $\quad\quad$ **for all** $f'' \in F' \setminus f'$ **do**
11. $\quad\quad\quad F'' = F' \setminus f''$
12. $\quad\quad\quad Q'' = 0$
13. $\quad\quad\quad$ **for all** $f''' \in F''$ **do**
14. $\quad\quad\quad\quad Q'' = Q'' + simplicity(f)$
15. $\quad\quad\quad$ **end for**
16. $\quad\quad\quad$ **if** $Q'' \geq Q$ **then**
17. $\quad\quad\quad\quad F = F''; Q = Q''$
18. $\quad\quad\quad$ **end if**
19. $\quad\quad$ **end for**
20. $\quad$ **end if**
21. **end for**
22. **return** F

the same set of sensors. This makes this problem essentially a pattern subset selection problem, and we therefore focus on the selection of a subset of equations that reduces the redundancy of this equation set.

One source of redundancy comes from the possible presence of irrelevant terms in the equations. For example, given the two equations,

$$f_x(t) = 1.0 \cdot s_y(t)$$

and

$$f_x(t) = 1.0 \cdot s_y(t) + 0.00001 \cdot s_z(t)$$

the latter might produce a smaller error, but is worse in the sense that it includes s_z that does not contribute significantly to the modeling of s_x, and is likely to play a more important role when paired to another target sensor. In order to specify a preference for equations with relevant terms (such as the first example), we define a simplicity measure for equations that is based on the scalars c_y. Although defining such a measure can be done in various ways, including rather sophisticated statistical tests for the contribution of each term, we have opted for a more straightforward approach here. The following definition states how the simplicity of an equation depends on the scales of its parameters:

$$simplicity(f) = \frac{1}{\sum_{c \in f} |\log |c||}.$$

In other words, we prefer equations with scalars c closer to 1. We assume here that signals are all within farily similar domains, which is the case in our data. Furthermore, we prefer c values to be closer to 1 (i.e. $\log |c|$ close to 0), to indicate a direct dependence.

Given our set of candidate equations C, can we find a subset $F \subseteq C$ such that it leads to the highest total simplicity? As the simplicity cannot be negative, we can trivially select all equations to achive a maximum simplicity. However, this would provide a lot of redundancy, as many of these equations will describe the same relationship between sensors. Therefore, we restrict this subset such that a relationship between s_x and s_y is described at most once. That is, for each pair of sensors (s_x, s_y) there is at most one equation f_x or f_y such that:

$$f_x = c_y \cdot s_y(t) + \ldots, \text{ or}$$
$$f_y = c_x \cdot s_x(t) + \ldots$$

In order to derive an interesting set of equations, one option is to evaluate all suitable subsets of C, and select that one that has the highest simplicity. How would this scheme perform?

For each set of selected equations, F, we need to check the simplicity for each $f \in F$ that can be done linearly in the length of f: $\mathcal{O}(L(f))$. For the complete set, this becomes $\mathcal{O}(|F| \cdot |S|)$. Furthermore, we need to check for every pair $(f_1, f_2) \in F \times F$ if it does not overlap: $\mathcal{O}(|S|^2)$. In total, this thus becomes for each set: $\mathcal{O}(|F| \cdot |S| + |F|^2 \cdot |S|^2) = \mathcal{O}(|F|^2 \cdot |S|^2)$ for each selected set. Clearly, there are $\mathcal{P}$(C) number of all possible subsets of equations. The total therefore becomes $\mathcal{O}(\mathcal{P}(C) \cdot (|F|^2 \cdot |S|^2))$. Clearly, an exhaustive search is far from feasible. Typically, $\mathcal{P}(C)$ would be by far the biggest factor in this equation, making it the main target to minimise.

Our alternative therefore is to perform a heuristic search through the C search space. In this search, we utilise a forward selection scheme in which we evaluate the candidate equations in order and select only those that contribute to the total simplicity (see Algorithm 1).

In our approach, we order the set of candidate equations, denoted by $\downarrow C$, on length either:

- ascending: $f_1 > f_2 \leftrightarrow L(f_1) > L(f_2)$, or
- descending $f_1 < f_2 \leftrightarrow L(f_1) > L(f_2)$.

Each candidate equation is then added to F and checked whether this increases the overall simplicity. After all candidates are evaluated, the resulting F is returned.

Due to overlap, a new candidate equation might not be considered for inclusion in the resulting set while it might provide with a good simplicity increase. Alternatively, we can then apply a pruning strategy on F. That is, we can check for every candidate pattern whether some of the already selected equations can be removed from the set (see Algorithm 2).

Table 1: Characteristics of the 145 used sensors.

sensor type	*#sensors*	location X-axis
1: geophone Z-axis	34	$\{1, 4\}$
2b: strain X-axis embedded	16	$\{6, 7\}$
2p: strain X-axis attached	34	$\{0, 2, 3, 4, 5\}$
3b: strain Y-axis embedded	28	$\{6, 7\}$
3p: strain Y-axis attached	13	$\{3, 5\}$
4b: temperature embedded	10	$\{7\}$
4p: temperature attached	10	$\{5\}$

4. EXPERIMENTS

In our experiments, we have used sensor measurement data derived from the *Hollandse Brug* in the Netherlands, as part of the InfraWatch project. This setup consist of 145 sensors. As can be seen in Table 1, there are 7 basic sensor types for which one a priori can expect that members to behave similarly. Therefore, in our first experiments, we have selected one target sensors from each of the 7 types.

In our preliminary experiments, we have aquired data representing 5 minutes of measurements, taken on the 24th of October in 2008, of all 145 sensors at 100 Hz, leading to 50 Mb of data. In our first experiments, we have downsampled this file with averaging to 1Hz, resulting in 297 distinct records.

Given each distinct target sensor, we use this dataset and Lagramge to fit equations on the target sensor. An equation grammar was used that produces linear functions of the form $f_x(t) = c_0 + \sum c_y \cdot s_y(t)$, as discussed. Note that this grammar can be easily upgraded to more complex, higher-order equations, were one so inclined. The amount of data available, and the expected nature of relationships between sensors plays a role in this decision also.

In our experiments, we have set the maximal prediction error to $1.0 \cdot 10^{-5}$. Unless reported otherwise, we have limited the search depth to 6, that is, at most 5 sensors and one constant c_0 can appear in the equation. Using a heuristic sum squared error-based beam search, we then obtain the 1000 best equations for each sensor type. In total, we therefore have 7000 candidate equations. In Table 2, examples are shown of the kinds of equation sets discovered, for two sensors: s_{100} and s_{301}. Also reported for each equation is the sum squared error that is obtained when approximating the target sensor.

4.1 Regular Equations

The candidate set of equations is ordered at the start of our forward selection algorithm. In our experiments we applied an order based on the length of the equations, either ascending or descending. With this forward selection scheme, we obtain much smaller equation sets out of the original candidate set. That is, sets with either 193 or just 1 equation, respectively, and with respect to the candidate set, we obtain reduction ratios of 2.8% and 0.014%. While these reductions seem very good, we should also focus on the resulting simplicity of the sets. In order to make this assesment, we take a look at the total simplicity, the sum of all simplicity values, of the resulting equation set. We see that we obtain a total simplicity of $48.1 \cdot 10^3$ and 105, for the ascending and descending order repectively. Moreover, we can have a look at how much of the sensor system is covered by these equation sets, which is 174 sensor pairs and 5 sensor pairs respectively.

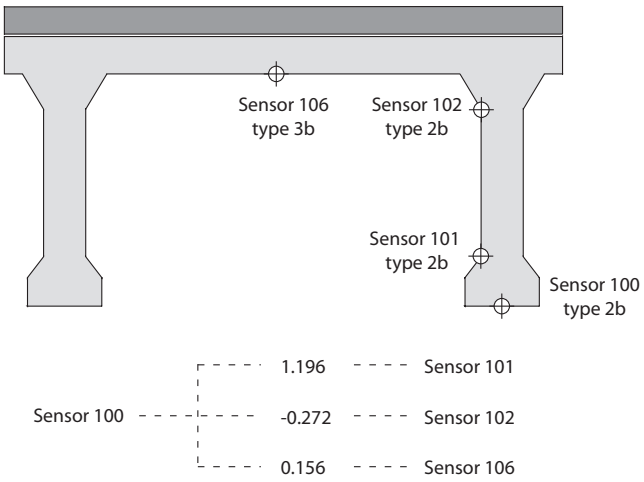


Figure 3: An example equation set shown in situ of the Hollandse Brug, please refer to Table 1 for the sensor type description.

However, such a forward selection algorithm with an overlap restriction is likely to select equations early on in the search process that might conflict with better candidates later on. Therefore, we have applied the pruning strategy for both candidate orders to observe its results.

As for the simplicity, we see that pruning leads to much better results in both the ascending and descending case. In Figure 5 (left), we depict the increase of simplicity during the run of the algorithm. As more candidates are being evaluated, we see that some of them can be added successfully to the equation set to increase its simplicity. Both orders show a similar increase in simplicity, although the descending approach leads to a slightly higher simplicity. The maximum obtained qualities are $1.57 \cdot 10^6$ and $1.72 \cdot 10^6$, for ascending and descending respectively.

We see the effect of pruning more clearly when looking at how the *cover* behaves over time (Figure 5 (right)). By cover, we mean the total number of unique sensors appearing in the right-hand side of the selected equations. Depicted for both candidate orders, we see how pruning affects the cover in a non-monotonic manner in favour of increasing the simplicity of the complete set of equations. In this respect, the descending-ordered candidate set gradually leads to larger covers of the sensor system, while the ascending order seems to peak early on in the search process. This eventually results in a cover of 40 and 154 for ascending and descending respectively.

We depict an example equation set in Figure 3. This result is obtained when using pruning on the descending-ordered candidate set as described earlier on. We show the sensors that are related to sensor 100, namely 101, 102, and 106:

$$f_{100}(t) = 23.30 + 1.196 \cdot s_{101}(t) - 0.272 \cdot s_{102}(t) + 0.156 \cdot s_{106}(t)$$

Note that all selected sensors fall within the same segment of the complete bridge (a total of 7 segments). Furthermore, all sensors are of the embedded type, which indicates that

Table 2: Examples of the first 5 equations found by Lagramge for sensors 100 and 220.

$$f_{100} = -4.799 + 1.16 \cdot s_{101} - 0.2364 \cdot s_{102} + 0.1773 \cdot s_{104} + 0.1104 \cdot s_{108} - 268.8 \cdot s_{233} \quad (SSE = 0.1084)$$
$$f_{100} = 16.98 + 1.143 \cdot s_{101} - 0.2896 \cdot s_{102} + 0.1347 \cdot s_{104} + 0.1221 \cdot s_{106} + 166.0 \cdot s_{223} \quad (SSE = 0.1086)$$
$$f_{100} = 16.70 + 1.14 \cdot s_{101} - 0.293 \cdot s_{102} + 0.1335 \cdot s_{104} + 0.1204 \cdot s_{106} + 128.8 \cdot s_{207} \quad (SSE = 0.109)$$
$$f_{100} = 11.8 + 1.150 \cdot s_{101} - 0.3028 \cdot s_{102} + 0.1469 \cdot s_{104} + 0.09248 \cdot s_{106} - 84.352 \cdot s_{219} \quad (SSE = 0.1092)$$
$$f_{100} = -4.812 + 1.152 \cdot s_{101} - 0.2431 \cdot s_{102} + 0.1916 \cdot s_{104} + 0.1069 \cdot s_{108} - 55.15 \cdot s_{201} \quad (SSE = 0.1094)$$
...
$$f_{301} = 2.215 - 4.291 \cdot 10^{-6} \cdot s_{105} + 3.502 \cdot s_{235} + 0.7977 \cdot s_{317} \quad (SSE = 1.888 \cdot 10^{-5})$$
$$f_{301} = 2.275 - 4.215 \cdot 10^{-6} \cdot s_{105} + 4.795 \cdot s_{236} + 0.7916 \cdot s_{317} \quad (SSE = 1.892 \cdot 10^{-5})$$
$$f_{301} = 2.251 - 4.206 \cdot 10^{-6} \cdot s_{105} + 3.273 \cdot s_{241} + 0.7939 \cdot s_{317} \quad (SSE = 1.895 \cdot 10^{-5})$$
$$f_{301} = 2.289 - 4.199 \cdot 10^{-6} \cdot s_{105} + 2.09 \cdot s_{233} + 0.7902 \cdot s_{317} \quad (SSE = 1.901 \cdot 10^{-5})$$
$$f_{301} = 2.328 - 4.2240 \cdot 10^{-6} \cdot s_{105} - 0.8519 \cdot s_{224} + 0.7865 \cdot s_{317} \quad (SSE = 1.902 \cdot 10^{-5})$$
...

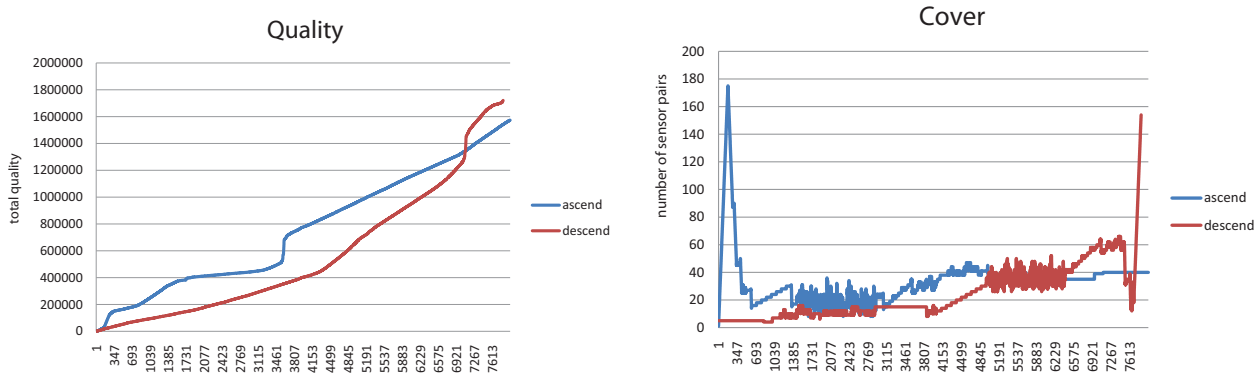


Figure 5: The increase of the simplicity (left) and the cover (right) of the equation set, both as a function of the evaluated candidate equations when using pruning.

this type of signal differs from the attached type in terms of physical characteristics.

When looking at the original signals, we see that sensor 102 is indeed an inverted signal opposed to sensor 100, and that 101 is quite similar to 100, and 106 is indeed also positively correlated. All signals show peaks occuring at similar times, which are our events of interest (see Figure 4). In this setup, we would select sensor 100 as a prototype to demonstrate which events are occurring in the sensor system.

4.2 Differential Equations

Apart from the regular algebraic grammar used in the previous section, we can use more elaborate grammars in order to fit more complex equations on the data. For example, we can use differential equations to approximate the target signal. This can actually be a very good way to model time dependent aspects of the sensor system, as it includes the temporal aspect more directly in the equations. To this end, we have also used the above sensor data to fit the following type of equation. Again, we have opted for relatively simple, though differential, models of the data:

$$\frac{df_x}{dt} = c_0 + \sum c_y \cdot s_y(t)$$

Like before, we have used a beam search with the same parameters to find the 1000 best equations that minimise the prediction error for the target sensor. The first five differential equations found are shown in Table 3. For equation selection on the differential equations, we see similar results as for the linear case. For ascending and descending, the forward selection without pruning leads to a (relatively small) simplicity of 983.2 and 27.6, a cover of 36 and 6, and an equation subset size of 35 and 2, respectively. As a demonstration, the two equations in this last subset are as follows:

$$\frac{df_{100}}{dt} = 128.7 - 0.40 \cdot s_{162}(t) + 5551 \cdot s_{240}(t) - 13.7 \cdot s_{318}(t)$$
$$\frac{df_{100}}{dt} = 32.8 + 0.12 \cdot s_{116}(t) - 1316 \cdot s_{209}(t) - 3.34 \cdot s_{317}(t)$$

Again, we see much better results when applying the pruning strategy on the candidate equation set. For ascending and descending, the forward selection with pruning leads to a simplicity of $7.57 \cdot 10^5$ and $1.39 \cdot 10^6$, a cover of 26 and 56, and a equation subset size of 10 and 45, respectively.

From this we can conclude that in terms of the number of selected equations in both cases we obtain very good reduction ratios, 0.5% and 0.02% without pruning, and 0.14% and 0.64% with pruning.

4.3 Select Target Sensors

In the previous experiments, we have used a set of 7 sensors, one for each type, as targets for which equations were

Table 3: Examples of the first 5 differential equations found by Lagramge for sensor 100.

$$\frac{df_{100}(t)}{dt} = 12.95 + 0.1115 \cdot s_{116}(t) - 1374.8 \cdot s_{209}(t) - 1.6136 \cdot s_{318}(t) \quad (SSE = 9.418)$$
$$\frac{df_{100}(t)}{dt} = -1.540 + 0.1076 \cdot s_{116}(t) - 1260.9 \cdot s_{209}(t) \quad (SSE = 9.43133)$$
$$\frac{df_{100}(t)}{dt} = 27.61 + 0.1614 \cdot s_{117}(t) - 1147.1 \cdot s_{209}(t) - 2.7610 \cdot s_{317}(t) \quad (SSE = 9.442)$$
$$\frac{df_{100}(t)}{dt} = 27.64 + 0.1682 \cdot s_{117}(t) - 1359.4 \cdot s_{209}(t) - 3.168 \cdot s_{318}(t) \quad (SSE = 9.483)$$
$$\frac{df_{100}(t)}{dt} = 16.63 + 0.1458 \cdot s_{117}(t) - 1099.2 \cdot s_{209}(t) - 1.787 \cdot s_{319}(t) \quad (SSE = 9.50)$$
...

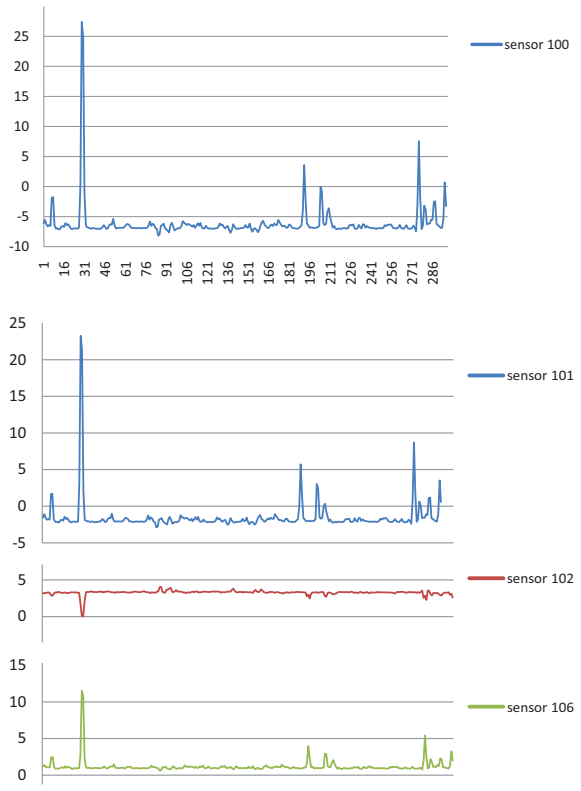


Figure 4: The sensor measurements for the signals that correspond to the example shown in Figure 3.

fitted. However, it could be that this background knowledge steers the search process significantly in a biased direction. Therefore, we have derived a set of candidate equations for each sensor in the system as target.

Moreover, we have reduced the search depth of Lagramge such that exactly one sensor can be paired with one target sensor. This allows the search space to be kept reasonable, while it still leads to over 20 thousand candidate equations. In addition, having only one sensor in the equation shows more clearly the direct relation between the sensors.

For this, we have used the best performing variant of our algorithm: with pruning and a descending-ordered candidate set. This resulted in an equation set of size 223. As for the target sensors, a set of 26 distinct sensors were selected as representations for the complete system. The resulting equation set had a simplicity of $3.74 \cdot 10^7$ and covered 196 pairs of the sensor system.

How does this relate to the case of hand-picking a set of target sensors? When selecting a set of target sensors from the complete sensor space, we see that not all sensor types are represented as targets. If we break these down to sensor type, we get the following distribution:

type	description	targets	equations
1	geophone Z-axis	8	12
2b	strain X-axis embedded	0	0
2p	strain X-axis attached	1	1
3b	strain Y-axis embedded	1	1
3p	strain Y-axis attached	0	0
4b	temperature embedded	6	14
4p	temperature attached	10	195

When visually inspecting the sensor signals, we see that those sensor types that are not so well described by other sensors tend to have more prototypes in the resulting set. For example, when inspecting the signals corresponding to sensor type 1, the geophone sensors, we see that all chosen target sensors of this type fluctuate quite a lot (see Figure 6).

5. CONCLUSION

Increasingly, physical systems are being equipped with sensors that in some form monitor its behaviour. In this paper, we focus on one such system in particular, the *Hollandse Brug*, which in the context of the *InfraWatch* project has been equipped with a multitude of sensors that measure aspects such as vibrations and strain. The aim of the project is to use the derived stream of sensor data to find interesting patterns and features that can be considered characteristic for its short and long-term behaviour.

In this paper, the focus was on finding a small set of interesting sensors within the large set of available sensors. As monitoring the complete system at once is very hard, we have proposed to model dependencies between sensors, and find characteristic sensors that can serve as a prototype for other sensors. The monitoring and exploratory analysis of the data can then be restricted to this subset of sensors. The prototype sensors should ideally contain the same features (peaks, response to traffic, etc.) as all the other sensors they represent.

Dealing with numeric data is a hard task for pattern mining techniques. However, we propose to shift the context slightly. Instead of grouping sensors based on their discrete events, we propose to group those sensors that can

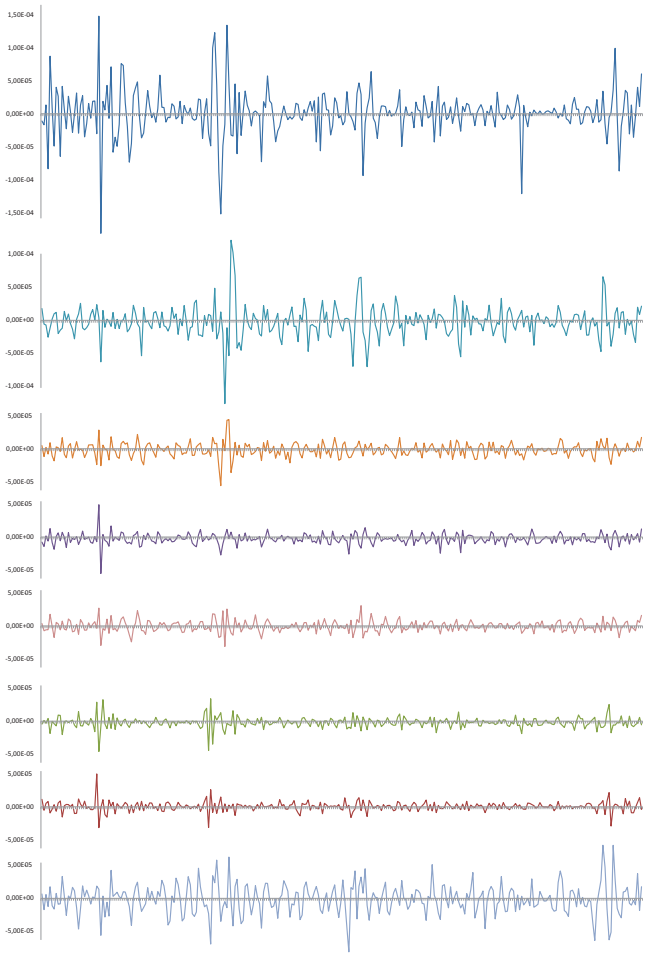


Figure 6: The signal values of some of the geophone sensors, that are hard to group. As expected, in contrast to other sensors, we see that the events are not as clearly present in the signals

be used well to describe other sensors in the form of equations. Although many specific implementations can be used, our inital results focus on linear and first-order differential equations. In these first experiments, we see that adjacent sensors, those that are likely to measure similar events, are indeed grouped by our approach.

When looking at the signal behaviour within the group we see that distinct peaks, which indicate distinct traffic loads, occur at similar time points, and stand out clearly from the noise in the signal.

Acknowledgements

The InfraWatch project is funded by the Dutch funding agency STW, under project number 10970. Access to sensor data of the Hollandse Brug is kindly provided by Strukton b.v.. Specifically Bas Obladen, Carlos Bosma and Hessel Galenkamp from Strukton were instrumental in setting up the sensors and data acquisition facilities, as well as arranging access to the data.

6. REFERENCES

[1] Bringmann, B., Zimmermann, A. *The Chosen Few: On Identifying Valuable Patterns*, In Proceedings ICDM 2007.

[2] M. Dejori, H.H. Malik, F. Moerchen, N.C. Tas, and C. Neubauer, 2009 *Development of Data Infrastructure for the Long Term Bridge Performance Program*, In Proceedings of Structures '09, Austin, USA.

[3] E. Doupal, R. Calderara, 2004, *Weigh-In-Motion*, In Proceedings of First International Conference on Virtual and Remote Weigh Stations, Orlando.

[4] Džeroski, S. and Todorovski, L. (1993) Discovering dynamics. In Proc. Tenth International Conference on Machine Learning, pages 97-103. Morgan Kaufmann, San Mateo, CA.

[5] Džeroski, S. and Todorovski, L. (1995) Discovering dynamics: from inductive logic programming to machine discovery. Journal of Intelligent Information Systems, 4: 89-108.

[6] Enders, W. *Applied Econometric Time Series.* John Wiley and Sons 2003.

[7] Knobbe, A., Blockeel, H., Koopman, A., Calders, T., Obladen, B., Bosma, C., Galenkamp, H., Koenders, E., Kok, J. *InfraWatch: Data management of large systems for monitoring infrastructural performance*, In Proceedings IDA 2010, Tucson, USA, 2010

[8] Knobbe, A. *Multi-Relational Data Mining.* IOS Press, Amsterdam, 2006. http://www.kiminkii.com/thesis.pdf.

[9] Knobbe, A., Ho, E.K.Y. *Maximally-Informative k-Itemsets, and their Efficient Discovery*, In Proceedings KDD 2006, 2006.

[10] A. Knobbe, 1997, *Data Mining for Adaptive System Management*, In Proceedings of PAKDD '97, London.

[11] A. Knobbe, Bart Marseille, Otto Moerbeek, Daniël M.G. van der Wallen, *Results in Adaptive System Management*, Benelearn'98

[12] G. Meijer, *Smart Sensor Systems*, 2008, ISBN: 978-0-470-86691-7, Hardcover, 404 pages.

[13] Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T. *Numerical Recipes.* Cambridge University Press, Cambridge, MA, 1986.

[14] Riggelsen, C., Ohrnberger, M., Scherbaum, F. *Dynamic Bayesian Networks for Real-Time Classification of Seismic Signals*, In Proceedings of PKDD '07.

[15] Todorovski, L. and Džeroski, S. (1997) Declarative bias in equation discovery. In Proc. Fourteenth International Conference on Machine Learning, pages 376-384. Morgan Kaufmann, San Mateo, CA.

Author Index