MINING PREDICTIVE PATTERNS AND EXTENSION TO MULTIVARIATE TEMPORAL DATA

by

Iyad Batal

BS, University of Damascus, 2005

MS, University of Pittsburgh, 2008

Submitted to the Graduate Faculty of

the Kenneth P. Dietrich School of

Arts and Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy in **Computer Science**

University of Pittsburgh

2012

UNIVERSITY OF PITTSBURGH

COMPUTER SCIENCE DEPARTMENT

This dissertation was presented

by

Iyad Batal

It was defended on

October 29, 2012

and approved by

Milos Hauskrecht, PhD, Associate Professor, Computer Science

Rebecca Hwa, PhD, Associate Professor, Computer Science

G. Elisabeta Marai, PhD, Assistant Professor, Computer Science

Jeff Schneider, PhD, Associate Research Professor, Computer Science (Carnegie Mellon University)

Dissertation Director: **Milos Hauskrecht, PhD**, Associate Professor, Computer Science

MINING PREDICTIVE PATTERNS AND EXTENSION TO MULTIVARIATE TEMPORAL DATA

Iyad Batal, PhD

University of Pittsburgh, 2012

An important goal of knowledge discovery is the search for patterns in the data that can help explaining its underlying structure. To be practically useful, the discovered patterns should be novel (unexpected) and easy to understand by humans. In this thesis, we study the problem of mining patterns (defining subpopulations of data instances) that are important for predicting and explaining a specific outcome variable. An example is the task of identifying groups of patients that respond better to a certain treatment than the rest of the patients.

We propose and present efficient methods for mining predictive patterns for both atemporal and temporal (time series) data. Our first method relies on frequent pattern mining to explore the search space. It applies a novel evaluation technique for extracting a small set of frequent patterns that are highly predictive and have low redundancy. We show the benefits of this method on several synthetic and public datasets.

Our temporal pattern mining method works on complex multivariate temporal data, such as electronic health records, for the event detection task. It first converts time series into time-interval sequences of temporal abstractions and then mines temporal patterns backwards in time, starting from patterns related to the most recent observations. We show the benefits of our temporal pattern mining method on two real-world clinical tasks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

PREFACE

During my Ph.D, I have received support from a number of people without whom the completion of this thesis would not be possible.

First of all, I would like to express my deepest gratitude to my advisor, Dr. Milos Hauskrecht, who introduced me to the fields of machine learning and data mining and taught me how to conduct high-quality research. I was privileged to work with Dr Greg Cooper, who always provided me with useful critiques to my research ideas. I would also like to thank my thesis committee, Dr. Rebecca Hwa, Dr Liz Marai and Dr Jeff Schneider for their valuable feedback and discussions during my thesis defense.

I want to thank our post-doc Hamed Valizadegan, with whom I worked during my last year of PhD. Also I want to thank the other members of Milos' machine learning lab: Saeed Amizadeh, Michal Valko, Quang Nguyen, Charmgill Hong and Shuguang Wang. Besides my studies, I am grateful for having nice friends in Pittsburgh, who made my stay very enjoyable. In particular, I want to mention Carolynne Ricardo, Rakan Maddah, John Fegali and Wissam Baino. I would also like to thank my dear friends from Syria, especially Fareed Hanna, Feras Meshal, Joseph Ayoub, Feras Deeb, Faten Fayad, Kinda Ghanem and Rami Batal.

Finally, I am indebted to my family for their unlimited and unconditional encouragement, support, and love. In particular, I'm very thankful to my loving parents George and May, my brother Ibrahim, my sister in law Hanady and my lovely niece Yara.

Thank you all!

1.0 INTRODUCTION

The large amounts of data collected today provide us with an opportunity to better understand the behavior and structure of many natural and man-made systems. However, the understanding of these systems may not be possible without automated tools that enable us to explore, explain and summarize the data in a concise and easy to understand form. Pattern mining is the field of research that attempts to discover patterns that describe important structures and regularities in data and present them in an understandable form for further use.

1.1 SUPERVISED PATTERN MINING

In this thesis, we study the application of pattern mining in the *supervised* setting, where we have a specific class variable (the outcome) and we want to find patterns (defining subpopulations of data instances) that are important for explaining and predicting this variable. Examples of such patterns are: "subpopulation of patients who smoke and have a positive family history are at a significantly higher risk for coronary heart disease than the rest of the patients", or "the unemployment rate for young men who live in rural areas is above the national average".

Finding predictive patterns is practically important for discovering "knowledge nuggets" from data. For example, finding a pattern that clearly and concisely defines a subpopulation of patients that respond better (or worse) to a certain treatment than the rest of the patients can speed up the validation process of this finding and its future utilization in patient-management. Finding predictive patterns is also important for the classification

task because the mined patterns can be very useful to predict the class labels for future instances.

In order to develop an algorithm for mining predictive patterns from data, we need to define a *search* algorithm for exploring the space of potential patterns and a *pattern selection* algorithm for choosing the "most important" patterns.

To search for predictive patterns, we use *frequent pattern mining*, which examines all patterns that occur frequently in the data. The key advantage of frequent pattern mining is that it performs a more complete search than other greedy search approaches, such as sequential covering [Cohen, 1995, Cohen and Singer, 1999, Yin and Han, 2003] and decision tree [Quinlan, 1993]. Consequently, it is less likely to miss important patterns. However, this advantage comes with the following disadvantages: 1) frequent pattern mining often produces a very large number of patterns, 2) many patterns are not important for predicting the class labels and 3) many patterns are redundant because they are only small variations of each other. These disadvantages greatly hinder the discovery process and the utilization of the results. Therefore, it is crucial to devise an effective method for selecting a small set of predictive and non-redundant patterns from a large pool of frequent patterns.

Most existing approaches for selecting predictive patterns rely on a quality measure (cf [Geng and Hamilton, 2006]) to score each pattern individually and then select the top scoring patterns [Nijssen et al., 2009, Bay and Pazzani, 2001, Li et al., 2001b, Brin et al., 1997a, Morishita and Sese, 2000]. In this thesis, we argue that this approach is ineffective and can lead to many spurious patterns. To overcome this shortcoming, we propose the *Minimal Predictive Patterns* (*MPP*) framework. This framework applies Bayesian statistical inference to evaluate the quality of the patterns. In addition, it considers the relations between patterns in order to assure that every pattern in the result offers a significant predictive advantage over all of its generalizations (simplifications).

We present an efficient algorithm for mining MPPs. As opposed to the commonly used approach, which first mines all frequent patterns and then selects the predictive patterns [Exarchos et al., 2008, Cheng et al., 2007, Webb, 2007, Xin et al., 2006, Kavsek and Lavrač, 2006, Deshpande et al., 2005, Li et al., 2001b], our algorithm integrates pattern selection with frequent pattern mining. This allows us to perform several strategies to prune the

search space and achieve a better efficiency.

1.2 TEMPORAL PATTERN MINING

Advances in data collection and data storage technologies have led to the emergence of complex *multivariate temporal datasets*, where data instances are traces of complex behaviors characterized by multiple time series. Such data appear in a wide variety of domains, such as health care [Hauskrecht et al., 2010, Sacchi et al., 2007, Ho et al., 2003], sensor measurements [Jain et al., 2004], intrusion detection [Lee et al., 2000], motion capture [Li et al., 2009], environmental monitoring [Papadimitriou et al., 2005] and many more. Designing algorithms capable of mining useful patterns in such complex data is one of the most challenging topics of data mining research.

In the second part of the thesis, we study techniques for mining multivariate temporal data. This task is more challenging than mining atemporal data because defining and representing temporal patterns that can describe such data is not an obvious design choice. Our approach relies on temporal abstractions [Shahar, 1997] to convert time series variables into *time-interval* sequences of abstract states and temporal logic [Allen, 1984] to represent temporal interactions among multiple states. This representation allows us to define and construct complex temporal patterns (time-interval patterns) in a systematic way. For example, in the clinical domain, we can express a concept like "the administration of heparin precedes a decreasing trend in platelet counts".

Our research work focuses primarily on mining predictive temporal patterns for *event detection* and its application to Electronic Health Records (EHR) data. For EHR data, each record (data instance) consists of multiple time series of clinical variables collected for a specific patient, such as laboratory test results and medication orders. The data also provide temporal information about the incidence of several adverse medical events, such as diseases or drug toxicities. Our objective is to mine patterns that can accurately predict adverse medical events and apply them to monitor future patients. This task is extremely used for intelligent patient monitoring, outcome prediction and decision support.

Mining predictive patterns in abstract time-interval data is very challenging mainly because the search space that the algorithm has to explore is extremely large and complex. All existing methods in this area have been applied in an unsupervised setting for mining temporal association rules [Moskovitch and Shahar, 2009, Wu and Chen, 2007, Winarko and Roddick, 2007, Papapetrou et al., 2005, Moerchen, 2006b, Höppner, 2003]. These methods are known to have a high computational cost and they do not scale up to large data.

In contrast to the existing methods, our work applies temporal pattern mining in the supervised setting to find patterns that are important for the event detection task. To efficiently mine such patterns, we propose the *Recent Temporal Patterns* (*RTP*) framework. This framework focuses the mining on temporal patterns that are related to most recent temporal behavior of the time series instances, which we argue are more predictive for event detection[1]. We present an efficient algorithm that mines time-interval patterns backward in time, starting from patterns related to the most recent observations. Finally, we extend the minimal predictive patterns framework to the temporal domain for mining predictive and non-spurious RTPs.

1.3 MAIN CONTRIBUTIONS

The main contributions of this thesis can be summarized as follows:

- **Supervised Pattern Mining:**
  - We propose the minimal predictive patterns framework for mining predictive and non-spurious patterns.
  - We show that our framework is able to explain and cover the data using fewer patterns than existing methods, which is beneficial for knowledge discovery.
  - We show that our mining algorithm improves the efficiency compared to standard frequent pattern mining methods.

[1] In the clinical domain, the most recent clinical measurements of a patient are usually more informative about his health state than distant measurements

- **Temporal Pattern Mining:**
  - We propose the recent temporal patterns framework to mine predictive patterns for event detection in multivariate temporal data.
  - We show that our framework is able to learn accurate event detection classifiers for real-world clinical tasks, which is a key step for developing intelligent clinical monitoring systems.
  - We show that our mining algorithm scales up much better than the existing temporal pattern mining methods.
  - We present the minimal predictive recent temporal patterns framework, which extends the idea of minimal predictive patterns to the temporal domain.

1.4 OUTLINE OF THE THESIS

This thesis is organized as follows. Chapter 2 outlines the related research in frequent pattern mining. Chapter 3 presents our approach for mining minimal predictive patterns. It also presents our experimental evaluations on several synthetic and benchmark datasets. Chapter 4 outlines the related research in temporal data mining. Chapter 5 presents our approach for mining predictive patterns in multivariate temporal data. It also presents our experimental evaluations on a synthetic dataset and on two real-world EHR datasets. Finally, Chapter 6 concludes the thesis.

Parts of this dissertation and closely related work were published in [Batal et al., 2012b, Batal et al., 2012a, Batal et al., 2012c, Batal et al., 2011, Batal and Hauskrecht, 2010b, Batal and Hauskrecht, 2010a, Batal et al., 2009, Batal and Hauskrecht, 2009]

2.0 FREQUENT PATTERN MINING

Frequent patterns are simply patterns that appear frequently in a dataset. These patterns can take a variety of forms such as:

1. **Itemset patterns**: Represent set of items [Agrawal et al., 1993, Yan et al., 2005, Cheng et al., 2007, Batal and Hauskrecht, 2010b, Mampaey et al., 2011].

2. **Sequential patterns**: Represent temporal order among items [Srikant and Agrawal, 1996, Zaki, 2001, Pei et al., 2001, Wang and Han, 2004].

3. **Time interval patterns**: Represent temporal relations among states with time durations [Höppner, 2003, Papapetrou et al., 2005, Winarko and Roddick, 2007, Moerchen, 2006a, Batal et al., 2009, Moerchen and Fradkin, 2010, Batal et al., 2011].

4. **Graph patterns**: Represent structured and semi-structured data such as chemical compounds [Kuramochi and Karypis, 2001, Vanetik et al., 2002, Yan and Han, 2002, Deshpande et al., 2005].

Frequent pattern mining plays an essential role for discovering interesting regularities that hold in data. Moreover, it has been extensively used to support other data mining tasks, such as classification [Wang and Karypis, 2005, Deshpande et al., 2005, Cheng et al., 2007, Batal and Hauskrecht, 2010b, Batal et al., 2011] and clustering [Agrawal et al., 1998, Beil et al., 2002].

Frequent pattern mining was first introduced by [Agrawal et al., 1993] to mine *association rules* for market basket data. Since then, abundant literature has been dedicated to this research and tremendous progress has been made.

In this chapter, we attempt to review the most prominent research on frequent pattern mining and focus mainly on **mining itemset patterns**[1]. Incorporating the temporal dimension in pattern mining is deferred to chapters 4 and 5.

The rest of this chapter is organized as follows. Section 2.1 provides some definitions that will be used throughout the chapter. Section 2.2 describes the most common frequent pattern mining algorithms. Section 2.3 reviews methods that attempt to reduce the number of frequent patterns (compress the results). Section 2.4 reviews methods that use patterns for supervised learning, where the objective is to mine patterns that predict well the class labels. Finally, Section 2.5 summarizes the chapter.

2.1 DEFINITIONS

Frequent pattern mining was first introduced by [Agrawal et al., 1993] for mining market basket data that are in **transactional** form. The goal was to analyze customer buying habits by finding associations between items that customers frequently buy together. For example, if a customer buys cereal, he is also likely to buy milk on the same trip to the supermarket. In this example, cereal and milk are called **items** and the customer's trip to the supermarket is called a **transaction**.

Formally, let $\Sigma = I_1, I_2, ..., I_n$ denotes the set of all items, which is also called the **alphabet**. An **itemset pattern** is a conjunction of items: $P = I_{q_1} \wedge ... \wedge I_{q_k}$ where $I_{q_j} \in \Sigma$. If a pattern contains k items, we call it a ***k-pattern*** (an item is a *1-pattern*). We say that pattern P is a **subpattern** of pattern P' (P' is a **superpattern** of P), denoted as $P \subset P'$, if every item in P is contained in P'. The **support** of pattern P in database D, denoted as $sup(P,D)$, is the number of instances in D that contain P. Given a user specified **minimum support** threshold σ, we say that P is **frequent pattern** if $sup(P,D) \geq \sigma$.

Example 1. *Consider the transaction data in Table 1, where the alphabet of items is* $\Sigma = \{A,B,C,D,E\}$ *and there are 5 transactions* T_1 *to* T_5 *(each represents a customer visit to the*

[1]Note that many of the techniques described in this chapter for itemset patterns are also applicable to more complex types of patterns.

supermarket). We can see that pattern $P = A \wedge C$ appears in transactions T_1, T_2 and T_4, hence the support of P is 3. If we set the minimum support $\sigma = 2$, then the frequent patterns for this example are: $\{A, C, D, E, A \wedge C, A \wedge D\}$.

Transaction	List of items
T_1	A, C, D
T_2	A, B, C
T_3	A, D, E
T_4	A, C
T_5	E

Table 1: An example of transaction data.

The original pattern mining framework was proposed to mine transaction data. However, the same concepts can be applied to relational **attribute-value** data, where each instance is described by a fixed number of attributes such as the data in Table 2.

Age	Education	Marital Status	Income
Young (< 30)	Bachelor	Single	Low ($< 50K$)
Middle age (30-60)	HS-grad	Married	Low ($< 50K$)
Middle age (30-60)	Bachelor	Married	Medium (50K-100K)
Senior (> 60)	PhD	Married	High ($> 100K$)

Table 2: An example of relational attribute-value data.

Attribute-value data can be converted into an equivalent transaction data if the data is *discrete*, which means the data contain only categorical attributes. In this case, we map each *attribute-value pair* to a distinct *item*. When the data contain numerical (continuous) attributes, these attributes should be *discretized* [Yang et al., 2005]. For example, the age attribute in Table 2 has been converted into three discrete values: *Young*, *Middle age* and *Senior*.

Table 3 shows the data in Table 2 in transaction format. Note that converting an attribute-value data into a transaction data ensures that all transactions have the same number of items (unless the original data contain missing values). After this transformation, we can apply pattern mining algorithms on the equivalent transaction data.

Transaction	List of items
T_1	Age=Young, Education=Bachelor, Marital Status=Single, Income=Low
T_2	Age=Middle age, Education=HS-grad, Marital Status=Married, Income=Low
T_3	Age=Middle age, Education=Bachelor, Marital Status=Married, Income=Medium
T_4	Age=Senior, Education=PhD, Marital Status=Married, Income=High

Table 3: The data in Table 2 in transaction format.

2.2 MINING ALGORITHMS

The task of pattern mining is challenging because the search space is very large. For instance, the search space of all possible itemset patterns for *transaction data* is *exponential in the number of items*. So if Σ is the alphabet of items, there are $2^{|\Sigma|}$ possible itemsets (all possible subsets of items). This search space can be represented by a lattice structure with the empty set at the bottom and the set containing all items at the top. Figure 1 shows the itemset lattice for alphabet $\Sigma = \{A, B, C\}$.

The search space of itemset patterns for *attribute-value data* is *exponential in the number of attributes*. So if there are d attributes and each attribute takes V possible values, there are $(V+1)^d$ valid itemsets. Note that the search space for more complex patterns, such as sequential patterns, time interval patterns or graph patterns, is even larger than the search space for itemsets.

Clearly, the naive approach to generate and count all possible patterns is infeasible. Frequent pattern mining algorithms make use of the minimum support threshold to restrict

the search space to a hopefully reasonable subspace that can be explored more efficiently. In the following, we describe the three main frequent pattern mining approaches: Apriori, pattern growth and vertical format.

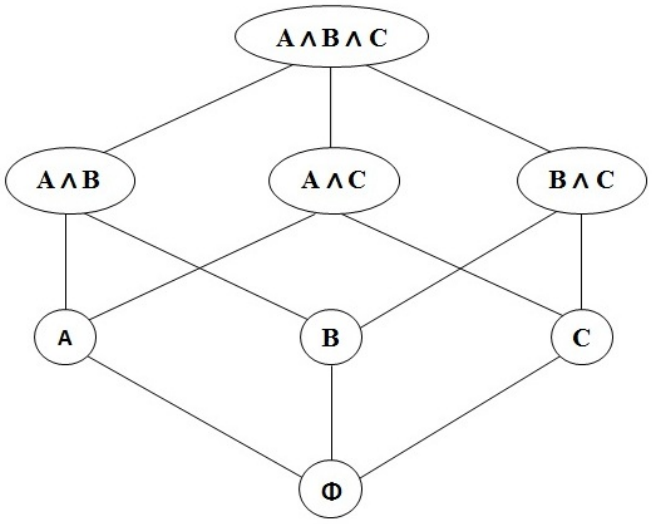


Figure 1: The itemset lattice for alphabet $\Sigma = \{A, B, C\}$.

2.2.1 The Apriori Approach

[Agrawal and Srikant, 1994] observed an interesting downward closure property among frequent patterns: *A pattern can be frequent only if all of its subpatterns are frequent*. This property is called the **Apriori property** and it belongs to a category of properties called *anti-monotone*, which means that if a pattern fail to pass a test, all of its superpatterns will fail the same test as well.

The Apriori algorithm employs an iterative *level-wise* search and uses the Apriori property to prune the space. It first finds all frequent items (*1-patterns*) by scanning the database and keeping only the items that satisfy the minimum support. Then, it performs the following two phases to obtain the frequent *k-patterns* using the frequent *(k-1)-patterns*:

1. *Candidate generation*: Generate candidate *k-patterns* using the frequent *(k-1)-patterns*. Remove any candidate that contains an infrequent *(k-1)-subpattern* because it is guar-

anteed not to be frequent according to the Apriori property.

2. *Counting*: Count the generated candidates and remove the ones that do not satisfy the minimum support.

This process repeats until no more frequent patterns can be found.

Example 2. *This example illustrates the candidate generation phase for itemset mining. Assume the algorithm found the following frequent 2-patterns:* $F_2 = \{A \wedge B, A \wedge C, B \wedge C, B \wedge D\}$. *One way to generate candidate k-patterns for itemset mining is by joining two (k-1)-patterns if they share the same* $k-2$ *prefix [Agrawal and Srikant, 1994]. Following this strategy, we join* $A \wedge B$ *with* $A \wedge C$ *to generate candidate* $A \wedge B \wedge C$. *Similarly, we join* $B \wedge C$ *with* $B \wedge D$ *to generate candidate* $B \wedge C \wedge D$. *However,* $B \wedge C \wedge D$ *is guaranteed not to be frequent because it contains an infrequent 2-subpattern:* $C \wedge D \notin F_2$. *Therefore,* $A \wedge B \wedge C$ *is the only candidate that survives the pruning.*

Since the Apriori algorithm was proposed, there have been extensive research on improving its efficiency when applied on very large data. These techniques include partitioning [Savasere et al., 1995], sampling [Toivonen, 1996], dynamic counting [Brin et al., 1997b] and distributed mining [Agrawal and Shafer, 1996]. Besides, Apriori has been extended to mine more complex patterns such as sequential patterns [Srikant and Agrawal, 1996, Mannila et al., 1997], graph patterns [Kuramochi and Karypis, 2001, Vanetik et al., 2002] and time interval patterns [Höppner, 2003, Moskovitch and Shahar, 2009, Batal et al., 2009].

2.2.2 The Pattern Growth Approach

Although the Apriori algorithm uses the Apriori property to reduce the number of candidates, it can still suffer from the following two nontrivial costs: 1) generating a large number of candidates, and 2) repeatedly scanning the database to count the candidates.

[Han et al., 2000] devised the **Frequent Pattern growth (FP-growth)** algorithm, which adopts a divide and conquer strategy and mines the complete set of frequent itemsets *without candidate generation*. The algorithm works by first building a compressed representation of the database called the Frequent Pattern tree (*FP-tree*). The problem of mining the database is transformed to that of mining the FP-tree.

Similar to Apriori, the algorithm starts by finding all frequent items. For each frequent item, the algorithm performs the following steps:

1. Extract the item conditional database.
2. Build the item conditional FP-tree.
3. Recursively mine the conditional FP-tree.

Pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from the conditional FP-tree.

[Han et al., 2000] showed that FP-growth is usually more efficient than Apriori. FP-growth has been extended to mine sequential patterns [Pei et al., 2001, Pei et al., 2007] and graph patterns [Yan and Han, 2002].

2.2.3 The Vertical Data Approach

Both Apriori and FP-growth mine frequent patterns from data represented in *horizontal format*, where every data instance represents a transaction and is associated with a list of items, such as the data in Table 1. Alternatively, the mining can be performed when the data is presented in **vertical format**, where every data instance is an item and is associated with a list of transactions, which is often called the ***id-list***. Table 4 shows the vertical format of the transaction data in Table 1. For example, the *id-list* of item C is $\{T_1, T_2, T_4\}$. Clearly, the support of an item is simply the length of its *id-list*.

Item	List of transactions
A	T_1, T_2, T_3, T_4
B	T_2
C	T_1, T_2, T_4
D	T_1, T_3
E	T_3, T_5

Table 4: The vertical data format of transaction data of Table 1.

[Zaki, 2000] proposed the *ECLAT* algorithm for mining frequent patterns using the vertical data format. Similar to Apriori, candidate *k-patterns* are generated from the frequent *(k-1)-patterns* using the Apriori property. However, instead of scanning the database to count every candidate, the algorithm computes the candidate's *id-list* by simply intersecting the *id-lists* of its *(k-1)-patterns*. For example, the *id-list* of pattern $A \wedge E$ in Table 4 is $\{T_1, T_2, T_3, T_4\} \bigcap \{T_3, T_5\} = \{T_3\}$, hence the support of $A \wedge E$ is 1. As we can see, the merit of this approach is that it does not have to scan the data to calculate the support of the candidates.

The vertical format approach has been extended to mine sequential patterns [Zaki, 2001] and time interval patterns [Batal et al., 2011].

2.3 CONCISE REPRESENTATIONS

One of the most serious disadvantages of frequent pattern mining is that it often produces a very large number of patterns. This greatly hinders the knowledge discovery process because the result is often overwhelming the user. Therefore, it is crucial to develop methods that can summarize (compress) the result in order to retain only the most "interesting" patterns. This section reviews some of the common techniques that aim to reduce the number of frequent patterns.

2.3.1 Lossless Compression

Lossless compression ensures that the result contains all information about the entire set of frequent patterns. A popular lossless representation is the *closed frequent patterns* [Pasquier et al., 1999], where a pattern P is a closed frequent pattern in dataset D if P is frequent in D and there is no proper superpattern P' such that P' has the same support as P. Several efficient algorithms have been proposed to mine frequent closed patterns [Zaki and Hsiao, 2002, Wang et al., 2003a].

Another lossless representation is the *non-derivable frequent patterns* [Calders and Goethals,

2002]. The idea is to derive a lower bound and an upper bound on the support of a pattern using the support of its subpatterns. When these bounds are equal, the pattern is called derivable. Therefore, we can mine only non-derivable patterns because they are sufficient to compute the support information for any frequent pattern. This idea was later extended to mine *non-derivable association rules* [Goethals et al., 2005].

2.3.2 Lossy Compression

Lossy compression usually provides greater compression rates than lossless compression, but looses some information about the frequent patterns. One of the earliest lossy representations is the *maximal frequent patterns* [Bayardo, 1998] [Yang, 2004], where a pattern P is a maximal frequent pattern in dataset D if P is frequent in D and there exists no proper superpattern of P that is also frequent in D. Note that by keeping only maximal frequent patterns, we can know the set of all frequent patterns. However, we loose the information about their exact support[2].

Another branch of lossless compression takes a **summarization** approach, where the aim is to derive k representatives that *approximate well the entire set of frequent patterns*. [Yan et al., 2005] proposed the *profile-based* approach to summarize a set of frequent patterns using representatives that cover most of the frequent patterns and are able to accurately approximate their support. These profiles are extracted using a generative model. The *Clustering-based* approach summarizes the frequent patterns by clustering them and selecting one representative pattern for each cluster. [Xin et al., 2005] defined the distance between two patterns in terms of the transactions they cover (two patterns are considered similar if they cover similar transactions). The patterns are clustered with a tightness bound δ to produce what they called δ-*clusters*, which ensures that the distance between the cluster representative and any pattern in the cluster is bounded by δ.

While the previous approaches [Yan et al., 2005, Xin et al., 2005] aim to find a set of patterns that *summarizes well all frequent patterns*, another view of this problem is to find a set of patterns that **summarizes well the dataset**. [Siebes et al., 2006] proposed a

[2]If we know that P is a maximal frequent pattern and we know its support, we cannot compute the exact support of its subpatterns.

formulation with the *Minimum Description Length* (*MDL*) principle. The objective is to find the set of frequent patterns that are able to compress the dataset best in terms of MDL. The authors showed that finding the optimal set is computationally intractable (an NP-hard problem) and proposed several heuristics to obtain an approximate solution. Recently, [Mampaey et al., 2011] proposed summarizing the data with a collection of patterns using a probabilistic *maximum entropy model*. Their method mines patterns iteratively by first finding the most interesting pattern, then updating the model, and then finding the most interesting pattern with respect to the updated model and so on.

2.3.3 Constraint-based Compression

A particular user may be only interested in a small subset of frequent patterns. **Constraint-based** mining requires the user to provide constraints on the patterns he would like to retrieve and tries to use these constraints to speed up the mining. Most of user constraints can be classified using the following four categories [Pei and Han, 2000]:

1. *Anti-monotone:* A constraint C_a is anti-monotone if and only if for any pattern that does not satisfy C_a, none of its superpatterns can satisfy C_a. For example, the minimum support constraint in frequent pattern mining is anti-monotone.
2. *Monotone:* A constraint C_m is monotone if and only if for any pattern that satisfies C_m, all of its superpatterns also satisfy C_m.
3. *Convertible:* A constraint C_c is convertible if it can be converted into an anti-monotone constraint or a monotone constraint by reordering the items in each transaction.
4. *Succinct:* A constraint C_s is succinct if we can explicitly and precisely enumerate all and only the patterns that satisfy C_s.

Example 3. *Suppose each item in the supermarket has a specific price and we want to impose constraints on the price of items in the patterns. An example of an anti-monotone constraint is sum(P.price)* $\leq \sigma$ *or min(P.price)* $\geq \sigma$*. An example of a monotone constraint is sum(P.price)* $\geq \sigma$ *or max(P.price)* $\geq \sigma$*. An example of a convertible constraint is avg(P.price)* $\geq \sigma$ *or avg(P.price)* $\leq \sigma$*. An example of a succinct constraint is min(P.price)* $\geq \sigma$ *or max(P.price)* $\leq \sigma$*.*

These different types of constraints interact differently with the mining algorithm:

1. Anti-monotone constraints can be pushed deep into the mining and can greatly reduce the search space.
2. Monotone constraints are checked for a pattern, and once satisfied, they do not have to be rechecked for its superpatterns.
3. Convertible constraints can be converted into anti-monotone or monotone constraints by sorting the items in each transaction according to their value in ascending or descending order [Pei and Han, 2000].
4. Succinct constraints can be pushed into the initial data selection process at the start of mining.

Constraint-based mining as described above considers *what the user wants, i.e., constraints, and searches for patterns that satisfy the specified constraints*. An alternative approach is to mine **unexpected patterns**, which considers *what the user knows, i.e., knowledge, and searches for patterns that surprise the user with new information*. [Wang et al., 2003b] defined a preference model which captures the notion of unexpectedness. [Jaroszewicz and Scheffer, 2005] proposed using a Bayesian network to express prior knowledge and defined the interestingness of a pattern to be the difference between its support in data and its expected support as estimated from the Bayesian network.

2.4 PATTERN MINING FOR SUPERVISED LEARNING

So far, we have discussed the main frequent pattern mining algorithms and described several methods for reducing the number of patterns. In this section, we turn our attention to methods that apply pattern mining in the supervised setting, where we have labeled training data of the form $D = \{x_i, y_i\}_{i=1}^{n}$ (y_i is the class label associated with instance x_i) and we want to mine patterns that can predict well the class labels for future instances.

In the supervised setting, we are only interested in rules that have the class label in their consequent. Hence, a rule is defined as $P \Rightarrow y$, where P (the condition) is a pattern and y is a class label. An example of a rule is *sky=cloudy* $\wedge$ *humidity=high* $\Rightarrow$ *play-tennis=No*.

In the following, we review several methods for supervised pattern mining (classification rule mining). We start by discussing methods from artificial intelligence and machine learning that try to achieve a similar goal. In particular, we discuss concept learning, decision tree induction and sequential covering. After that, we describe methods that use frequent pattern mining and contrast them to the other approaches.

2.4.1 Concept Learning

Concept learning is one of the most classical problems in artificial intelligence. The setting is that the learner is presented with training data of the form $D = \{x_i, c(x_i)\}_{i=1}^{n}$, where $c(x_i)$ is the concept associated with instance x_i. Instances for which $c(x_i) = 1$ are called *positive examples* (members of the target concept) and instances for which $c(x_i) = 0$ are called *negative examples* (nonmembers of the target concept). Let h denote a Boolean-valued function defined over the input space (h is called a *hypothesis*) and let H denote the space of all possible hypotheses the learner may consider. The problem faced by the learner is to find $h \in H$ such that $h(x) = c(x)$ for all x.

In concept learning, the hypothesis space H is determined by the human designer choice of hypothesis representation. Most commonly, H is restricted to include only conjunction of attribute values. For example, assume the data contain four attributes: *sky*, *temp*, *humidity* and *wind*. Hypothesis $h =< sky = ?, temp = hot, humidity = high, wind = ? >$ means that the target concept is true when the value of *temp* is *hot* and the value of *humidity* is *high* (regardless of the values of *sky* and *wind*). Note that *if we use conjunctive hypothesis space, the definition of a hypothesis becomes equivalent to the definition of an itemset pattern* (see Section 2.1). For example, hypothesis h is exactly the same as pattern $temp = cold \wedge humidity = high$. Hence, *the search space for learning conjunctive description hypotheses is the same as the search space of itemset mining for relational attribute-value data*.

A useful structure that is used for concept learning is the *general-to-specific* partial ordering of hypotheses. For example, hypothesis $h_1 = < sky = ?, temp = ?, humidity = high, wind = ? >$ is *more-general-than* $h_2 = < sky = clear, temp = warm, humidity = high, wind = ? >$.

Note that this is *exactly the definition of subpatterns*, where pattern h_1 is a subpattern of pattern h_2. The *general-to-specific* partial ordering is used to organize the search through the hypothesis space. In the following, we describe two common concept learning algorithms: find-S and candidate elimination.

Find-S finds the *most specific* hypothesis in H that is consistent with (correctly classifies) the training data. It starts with the most specific hypothesis (a hypothesis that does not cover any example) and generalizes this hypothesis each time it fails to cover a positive training example. This algorithm has many serious drawbacks. First, it is unclear whether we should prefer the most specific consistent hypothesis over, say the most general consistent hypothesis or some other hypothesis of intermediate generality [Mitchell, 1997]. Second, there is no way to determine whether *find-S* has found the only hypothesis in H consistent with the data (converged), or whether there are other hypotheses in H that are also consistent with the data.

To overcome these shortcomings, the *candidate elimination* algorithm was proposed by [Mitchell, 1982]. This algorithm outputs a description of the set of *all hypotheses* consistent with the training data, which is represented by the *version space*. The idea is to use the *more-general-than* partial order to represent the version space without explicitly enumerating all of its members. This is accomplished by storing only its most specific members (the *S-boundary*) and its most general members (the *G-boundary*). The algorithm incrementally refines the *S-boundary* and *G-boundary* as new training examples are encountered.

It is important to note that concept learning methods rely on two strong assumptions:

1. The hypothesis space H contains the true target concept: $\exists\, h \in H : h(x) = c(x)\ \forall x \in X$.
2. The training data contain no errors (noise free).

For instance, if the hypothesis space supports only conjunctive description and the true target concept is a disjunction of attribute values, then concept learning will fail to learn the concept. One obvious fix to this problem is to use a hypothesis space that is capable of representing every teachable concept (every possible Boolean function). Unfortunately, doing so causes the concept learning method to learn a concept that exactly fits the training data, hence totally fails to generalize to any instance beyond the training data [Mitchell,

1997]. In the remainder of this section, we describe methods that do not rely on these two assumptions.

2.4.2 Decision Tree Induction

Decision tree induction is a popular machine learning technique for building classification models. An example of a decision tree is shown in Figure 2. Each internal node in the tree denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label (predicts the concept *play-tennis* in this example). Many algorithms exist to learn a decision tree, such as ID3 [Quinlan, 1986], CART [Breiman et al., 1984] and C4.5 [Quinlan, 1993]. All of these algorithms build the decision tree from the root downward in a greedy fashion.

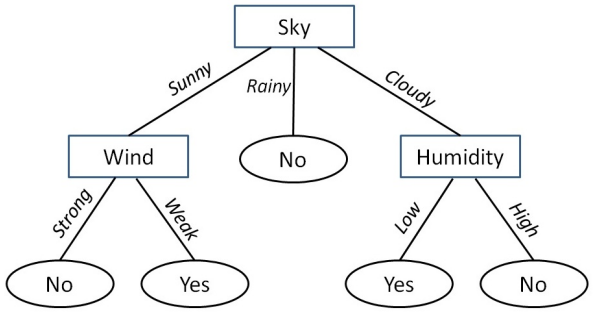


Figure 2: An example decision tree for the concept *play-tennis*.

One obvious way to obtain a set of classification rules is to first learn a decision tree, then translate the tree into an equivalent set of rules: one rule is created for each path from the root to a leaf node. That is, each internal node along a given path is added to the rule antecedent (with conjunction) and the leaf node becomes the rule consequent. For example, the rules corresponding to the tree in Figure 2 are:

- R_1 : *sky* = *sunny* $\wedge$ *wind* = *strong* $\Rightarrow$ *play-tennis* = *No*
- R_2 : *sky* = *sunny* $\wedge$ *wind* = *weak* $\Rightarrow$ *play-tennis* = *Yes*
- R_3 : *sky* = *rainy* $\Rightarrow$ *play-tennis* = *No*

- R_4 : *sky* = *cloudy* $\wedge$ *humidity* = *low* $\Rightarrow$ *play-tennis* = *Yes*
- R_5 : *sky* = *cloudy* $\wedge$ *humidity* = *high* $\Rightarrow$ *play-tennis* = *No*

Because every decision tree induces a partition of the input space, rules that are extracted directly from the tree are *mutually exclusive* and *exhaustive*. Mutually exclusive means that the rules do not overlap (an instance can be covered by only one rule), while exhaustive means that the rules cover the entire input space (every instance is cover by a rule).

There are several drawbacks for using rules from a decision tree. First, the extracted rules have a very restrictive form. For example, the attribute of the root note has to appear in every rule. Second, the rules are often difficult to interpret, especially when the original decision tree is large (the rules are often more difficult to interpret than the original tree). Finally, since the decision tree is built greedily, the resulting rules may miss important patterns in the data. To alleviate some of these problems, rules post-pruning can be applied as follows: for each rule, remove items from its antecedent if they do not improve its estimated performance [Quinlan, 1993]. Note that after performing rule post-pruning, the resulting rules will no longer be mutually exclusive and exhaustive.

2.4.3 Sequential Covering

Sequential covering learns a set of rules based on the strategy of learning one rule, removing the data it covers and then repeating the process. Sequential covering relies on the *learn-one-rule* subroutine, which accepts a set of positive and negative training examples as input and then outputs a single rule that tries to cover many of the positive examples and few of the negative examples.

learn-one-rule works by greedily adding the item (attribute-value pair) that most improves the rule's performance (e.g. the precision) on the training data. Once this item has been added, the process is repeated to add another item and so on until the rule achieves an acceptable level of performance. That is, *learn-one-rule* performs a greedy *general to specific* search by staring with the most general rule (the empty rule) and adding items to its antecedent to make it more specific. Note that this is the opposite of the *find-S* concept

learning algorithm (Section 2.4.1), which performs a *specific to general* search.

Sequential covering methods invoke *learn-one-rule* on all available training data, remove the positive examples covered by the rule, and then invoke it again to learn another rule based on the remaining training data and so on. The most common sequential covering algorithms are CN2 [Clark and Niblett, 1989], RIPPER [Cohen, 1995], SLIPPER [Cohen and Singer, 1999] and CPAR [Yin and Han, 2003]. Sequential covering has been extended by [Quinlan, 1990] to learn first-order rules (inductive logic programming), which are outside the scope of this thesis.

Let us now compare sequential covering rules and decision tree rules. Both approaches rely on a greedy search to explore the space of rules (patterns). However, the main difference is that sequential covering learns one rule at a time, while decision tree induction learns a set of rules simultaneously as part of a single search. To see this, notice that at each step of the search, a decision tree method chooses among alternative *attributes* by comparing the partitions of the data they generate, while a sequential covering method chooses among alternative *items* (attribute-value pairs) by comparing the subset of data they cover. In other words, the choice of a decision node in decision tree induction corresponds to choosing the precondition for multiple rules that are associated with that node (attribute). Therefore, decision tree usually makes fewer independent choices than sequential covering.

The main drawback of sequential covering is that it relies on many greedy choices: not only each rule is built greedily (using the *learn-one-rule* subroutine), but also the set of rules are obtained greedily (a single rule is learned at each iteration without backtracking). As with any greedy search, there is a danger of making a suboptimal choice at any step, which can affect the quality of the final results.

2.4.4 Frequent Patterns for Classification

As we discussed in Section 2.4.1, concept learning methods search an incomplete hypothesis space because they totally fail when the hypothesis space is complete (the learned concept would exactly replicate the training data). On the other hand, decision tree induction and sequential covering search the complete hypothesis space (i.e., a space capable of expressing

any discrete-valued function). However, the space is searched incompletely using greedy heuristics. In comparison, frequent pattern mining uses a complete hypothesis space and performs a more complete search than decision tree and sequential covering. The reason is that frequent pattern mining examines all patterns that occur frequently in the data instead of relying on greedy choices to explore the patterns.

Frequent patterns have been demonstrated to be useful for classification. Earlier approaches focused on **associative classification**, where rules describing strong associations between frequent patterns and class labels are used to build a *rule-based* classifier. In many studies, associative classification has been found to outperform some traditional classifiers, such as C4.5 decision trees [Quinlan, 1993]. *Classification Based Association* (*CBA*) [Liu et al., 1998] is the first associative classification method. It uses frequent pattern mining to mine a set of class association rules and uses the most confident (accurate) rule to classify test instances. *Classification based on Multiple Association Rules* (*CMAR*) [Li et al., 2001b] is more efficient than CBA because it applies several rule pruning strategies and uses a tree structure for efficient storage and retrieval of rules. In addition, CMAR can be more accurate than CBA because it considers multiple rules when making its class prediction (weighted majority voting) as opposed to using only a single rule as in CBA. [Cong et al., 2005] applies associative classification on gene expression profiles. Their method mines the top k covering rule groups for each instance and use them to construct the classifier. *HARMONY* [Wang and Karypis, 2005] uses an *instance-centric* approach to assure that for each training instance, one of the highest confidence rules covering the instance is included in the final set of rules. [Veloso et al., 2006] proposed *Lazy Associative Classification* (*LAC*), where the mining is defer until classification time. The advantage of LAC is that it restricts the search space by mining only rules that apply to the test instance. However, its disadvantage is that the mining is performed separately for each test instance, which becomes computationally expensive when there are many testing instances.

Recently, the focus shifted from associative classification to **pattern-based classification**, where discriminative frequent patterns are used to define *new features* in order to improve the performance of standard classification methods. [Cheng et al., 2007] conducted a systematic study to establish a connection between the support and several discriminative

measures, such as information gain and fisher score. They proposed using frequent patterns to represent the data in a different space, in which standard classifiers like SVM and C4.5 can be used to learn the model. Pattern-based classification has also been used to classify more complex structures, such as sequences [Tseng and Lee, 2005, Exarchos et al., 2008], graphs [Deshpande et al., 2005] and time interval sequences [Batal et al., 2009, Batal et al., 2011].

The most common approach for using frequent patterns for classification is to apply the **two-phase** approach, which mines *all* frequent patterns in the first phase and then selects the most discriminative patterns in the second phase [Cheng et al., 2007, Tseng and Lee, 2005, Exarchos et al., 2008, Deshpande et al., 2005]. In contrast, the works by [Fan et al., 2008, Cheng et al., 2008] attempt to *directly* mine discriminative patterns. The *Model Based Search Tree* (*MBST*) method [Fan et al., 2008] uses frequent pattern mining to build a *decision tree*. The basic idea is to partition the data in a top down manner and construct a tree as follows: At each node of the tree, 1) invoke a frequent pattern mining algorithm, 2) select the most discriminative pattern (according to information gain), 3) divide the data into two subsets, one containing this pattern and the other not, and 4) repeat the process recursively on the two subsets. The *Direct Discriminative Pattern Mining* (*DDPMine*) method [Cheng et al., 2008] is similar to [Fan et al., 2008] in that it mines the most discriminative patterns on progressively shrinking subsets of the data. However, DDPMine applies the *sequential covering* paradigm by mining the most discriminative frequent pattern (according to information gain), removing the instances covered by this pattern and recursively applying the algorithm on the remaining instances. DDPMine uses an upper bound on information gain (derived in [Cheng et al., 2007]) to prune parts of the search space that are guaranteed not to contain patterns with higher information gain than the current best pattern.

2.5 SUMMARY

Frequent pattern mining has been a focused theme in data mining research for over a decade. There have been hundreds of research publications, developments and application

activities in this domain. In this chapter, we did not attempt to provide a complete coverage of this topic, but we highlighted the aspects that are most relevant to this thesis. We mostly emphasized on two important research problems in frequent pattern mining: concise representations of frequent patterns and using pattern mining for supervised learning.

Several concise representation methods have been proposed for obtaining a compact but high quality set of patterns that are most useful for knowledge discovery. For most methods, the objective can be one of the following:

1. Obtain a lossless compression of all frequent patterns.
2. Obtain a "good" (but lossy) compression of all frequent patterns.
3. Obtain patterns that best summarize the data.
4. Obtain patterns that satisfy user constraints.
5. Obtain patterns that are surprising to the user (based on his prior knowledge).

Using pattern mining for supervised learning is a another interesting topic. Earlier approaches focused on concept learning, decision tree induction and sequential covering. In recent years, there has been a lot of research in data mining on using frequent pattern mining to improve classification performance. An important research direction is to develop more efficient pattern-based classification methods that can focus the search on predictive patterns instead of exploring the entire space of frequent patterns. We will address this issue in the next chapter.

3.0 MINING PREDICTIVE PATTERNS

Frequent Pattern Mining (FPM) is a very popular data mining technique for finding useful patterns in data. Since it was introduced by [Agrawal et al., 1993], FPM has received a great deal of attention and abundant literature has been dedicated to this research (see [Han et al., 2007]).

In this chapter, we study the application of pattern mining in the *supervised* setting, where we have a specific class variable (the outcome) and we want to find patterns (defining subpopulations of data instances) that are important for explaining and predicting this variable. These patterns are presented to the user in terms of if-then rules that are intuitive and easy to understand. Examples of such rules are: "If a patient smokes and has a positive family history, then he is at a significantly higher risk for lung cancer than the rest of the patients". This task has a high practical relevance in many domains of science or business. For example, finding a pattern that clearly and concisely defines a subpopulation of patients that respond better (or worse) to a certain treatment than the rest of the patients can speed up the validation process of this finding and its future utilization in patient-management.

We use FPM to explore the space of patterns because it performs a more systematic search than heuristic rule induction approaches, such as greedy sequential covering [Clark and Niblett, 1989, Cohen, 1995, Cohen and Singer, 1999, Yin and Han, 2003]. However, the disadvantage of FPM is that it often produces a large number of patterns. Moreover, many of these patterns are redundant because they are only small variations of each other. This large number of patterns (rules) easily overwhelms the domain expert and hinders the process of knowledge discovery. Therefore, it is crucial to devise an effective method for selecting a small set of predictive and non-redundant patterns from a large pool of frequent patterns.

To achieve this goal, we propose the *Minimal Predictive Patterns* (*MPP*) framework. This framework applies Bayesian inference to evaluate the quality of the patterns. In addition, it considers the structure of patterns to assure that every pattern in the result offers a significant predictive advantage over all of its generalizations (simplifications). We present an efficient algorithm for mining the MPP set. As opposed to the widely used two-phase approach (see Section 2.4.4), our algorithm integrates pattern selection and frequent pattern mining. This allows us to perform a lot of pruning in order to speed up the mining.

The rest of the chapter is organized as follows. Section 3.1 provides some definitions that will be used throughout the chapter. Section 3.2 describes the problem of supervised descriptive rule discovery. Section 3.3 describes the problem of pattern-based classification. Section 3.4 illustrates the problem of spurious patterns. Section 3.5 presents our approach for mining minimal predictive patterns. We start by defining a Bayesian score to evaluate the predictiveness of a pattern compared to a more general population (Section 3.5.1). Then we introduce the concept of minimal predictive patterns to deal with the problem of spurious patterns (Section 3.5.2). After that, we present our mining algorithm and introduce two effective pruning techniques (Section 3.5.3). Section 3.6 presents our experimental evaluation on several synthetic and publicly available datasets. Finally, Section 3.7 summarizes the chapter.

3.1 DEFINITIONS

We are interested in applying pattern mining in the *supervised* setting, where we have a special target variable Y (the class variable) and we want to find patterns that are important for describing and predicting Y. In this chapter, we focus on supervised pattern mining for relational *attribute-value* data $D = \{x_i, y_i\}_{i=1}^{n}$, where every instance x_i is described by a fixed number of attributes and is associated with a class label $y_i \in dom(Y)$. We assume that all attributes have discrete values (numeric attributes must be discretized [Fayyad and Irani, 1993, Yang et al., 2005]). As we discussed in Section 2.1, the data can be converted into an equivalent transactional format.

We call every attribute-value pair an **item** and a conjunction of items an **itemset pattern**, or simple a **pattern**. A pattern that contains k items is called a ***k-pattern*** (an item is a *1-pattern*). For example, *Education* = *PhD* $\wedge$ *Marital-status* = *Single* is a *2-pattern*.

Pattern P is a **subpattern** of pattern P', denoted as $P \subset P'$, if every item in P is contained in P' and $P \neq P'$. In this case, P' is a **superpattern** of P. For example, P_1 : *Education* = *PhD* is a subpattern of P_2 : *Education* = *PhD* $\wedge$ *Marital-status* = *Single*. The subpattern (*more-general-than*) relation defines a *partial ordering* of patterns, i.e. a **lattice structure**, as shown in Figure 3.

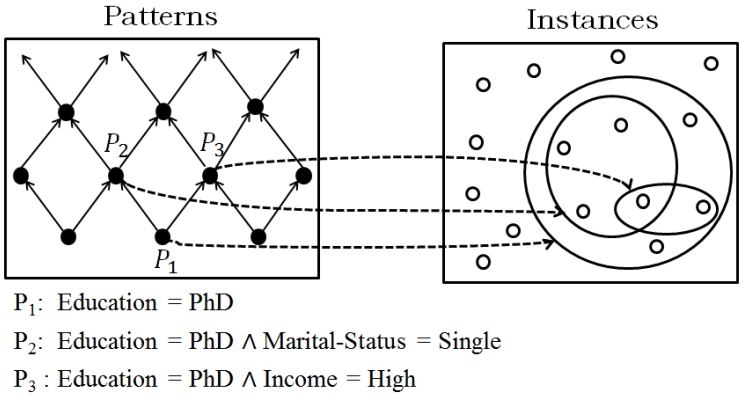


Figure 3: The box on the left shows the set of all patterns and the box on the right shows the set of all instances. Each pattern is associated with a group of instances that satisfy the pattern. The patterns are organized in a lattice structure according to the subpattern-superpattern relation.

Instance x_i satisfies pattern P, denoted as $P \in x_i$, if every item in P is present in x_i. Every pattern P defines a **group** (subpopulation) of the instances that satisfy P: $G_P = \{(x_i, y_i) : x_i \in D \ \wedge \ P \in x_i\}$. If we denote the empty pattern by ϕ, G_ϕ represents the entire data D. *Note that $P \subset P'$ (P is a subpattern of P') implies that $G_P \supseteq G_{P'}$* (see Figure 3).

The **support** of pattern P in dataset D, denoted as $sup(P, D)$, is the number of instances in D that satisfy P (the size of G_P). Given a user defined **minimum support** threshold σ, P is called a **frequent pattern** if $sup(P, D) \geq \sigma$.

Because we apply pattern mining in the supervised setting, we are only interested in

mining rules that predict the class variable. Hence, a **rule** is defined as $P \Rightarrow y$, where P (the condition) is a pattern and $y \in dom(Y)$ (the consequent) is a class label. We say that $P \Rightarrow y$ is a **subrule** of $P' \Rightarrow y'$ if $P \subset P'$ and $y = y'$.

A rule is usually assessed by its coverage and confidence. The **coverage** of $P \Rightarrow y$, denoted as $cov(P \Rightarrow y)$, is the proportion of instances in the data that satisfy P. The **confidence** of $P \Rightarrow y$, denoted as $conf(P \Rightarrow y)$, is the proportion of instances from class y among all the instances that satisfy P. By using D_y to denote the instances in D that belong to class y:

$$conf(P \Rightarrow y) = \frac{sup(P, D_y)}{sup(P, D)}$$

We can see that the confidence of $P \Rightarrow y$ is the *maximum likelihood estimation* of $Pr(Y = y|G_P)$. Intuitively, if pattern P is predictive of class y, we expect $conf(P \Rightarrow y)$ to be larger that the prior probability of y in the data.

3.2 SUPERVISED DESCRIPTIVE RULE DISCOVERY

Rule discovery is a very important tool for knowledge discovery because it has the advantage of representing the knowledge in terms of if-then rules that are easy to interpret by humans. This can facilitate the process of discovery and utilization of new practical findings.

Rule discovery using frequent pattern mining (i.e., association rule mining) has been mostly applied in the unsupervised setting to find rules that describe strong associations between different items.

In this work, we are interested in applying rule mining in the supervised setting. Our aim is to find a set of comprehensible rules/patterns that are statistically interesting compared to the entire data, e.g., the rules should have wide coverage and unusual distributional characteristics with respect to the class variable [Lavrač and Gamberger, 2005]. This task appeared in the literature under a variety of different names, such as contrast set mining [Bay and Pazzani, 2001], emerging pattern mining [Dong and Li, 1999, Bailey et al., 2002, Yu et al., 2012] and subgroup discovery [Lavrač and Gamberger, 2005, Kavsek and

Lavrač, 2006]. Later on, [Novak et al., 2009] provided a unifying framework of this work which is named **Supervised Descriptive Rule Discovery** (***SDRD***).

A straightforward approach to *SDRD* is to use a rule quality measure (cf [Geng and Hamilton, 2006]) to score each rule by contrasting it to the general population (the entire data) and report the top rules to the user [Nijssen et al., 2009, Bay and Pazzani, 2001, Li et al., 2001b, Brin et al., 1997a, Morishita and Sese, 2000]. We will argue that this approach is ineffective and can lead to many spurious rules. We start by illustrating this problem using an example and then describe it more formally in Section 3.4.

Example 4. *Assume our objective is to identify populations of patients who are at high risk of developing Coronary Heart Disease (CHD). Assume our dataset contains 150 instances, 50 of them are CHD cases and the others are controls. That is, the CHD prior in our data is 33.3%.*

Now, our task is to evaluate the following 5 rules:

- R_1: *Race = African American* $\Rightarrow$ *CHD*
  [#cases=19, #controls=40, conf=32.2%]
- R_2: *Race = Caucasian* $\Rightarrow$ *CHD*
  [#cases=32, #controls=58, conf=35.56%]
- R_3: *Family history = Yes* $\Rightarrow$ *CHD*
  [#cases=30, #controls=20, conf=60%]
- R_4: *Race = African American* $\wedge$ *Family history = Yes* $\Rightarrow$ *CHD*
  [#cases=11, #controls=8, conf=57.89%]
- R_5: *Race = Caucasian* $\wedge$ *Family history = Yes* $\Rightarrow$ *CHD*
  [#cases=21, #controls=11, conf=65.63%]

For each rule, we show the number of CHD cases and the number of controls that the rule covers. We also show the confidence of the rule.

The original association rule mining framework [Agrawal et al., 1993] outputs all rules with higher confidence than a user specified minimum confidence threshold. For instance, if the minimum confidence is 50%, rules R_3, R_4 and R_5 will be returned to the user.

One of the commonly used approaches to filter out uninteresting rules is to apply the χ^2 *test* to assure that there is a significant positive correlation between the condition and the consequent of each rule [Nijssen et al., 2009, Bay and Pazzani, 2001, Li et al., 2001b, Brin et al., 1997a, Morishita and Sese, 2000]. If we apply the χ^2 test on our rules, the *p-values* we get for R_1, R_2, R_3, R_4 and R_5 are 0.813, 0.479, 9.6×10^{-7}, 0.015 and 1.2×10^{-5}, respectively. That is, R_3, R_4 and R_5 are all statistically significant with respect to a significance level $\alpha = 0.05$. Moreover, these rules will be considered interesting using most rule quality measures [Geng and Hamilton, 2006].

The main problem with these approaches is that they *evaluate each rule individually without considering the relations between the rules*. For example, if we are given rule R_4 by itself, we may think it is an important rule. However, by looking at the other rules, we can see that R_4 is not interesting because it is more specific than R_3 (covers a smaller population) and has a lower confidence.

To tackle this problem, [Bayardo, 1999] proposed the *confidence improvement* constraint, which says that each rule in the result should have a higher confidence than all of its sub-rules:

$$conf(P \Rightarrow y) - \max_{S \subset P}\{\, conf(S \Rightarrow y)\,\} > 0$$

This filter have been used quite often in the rule mining literature [Grosskreutz et al., 2010, Webb, 2007, Li et al., 2001b, Li et al., 2001a]. If we applied the confidence improvement constraint to our working example, R_2, R_3 and R_5 will be retained.

As we can see, both the χ^2 test and the confidence improvement constraint agree that R_5 is an interesting rule. However, this may not be true and the observed improvement in the confidence of R_5 (65.63%) compared to the confidence of R_3 (60%) can be due to chance rather than actual causality. In fact, there is a high chance that refining a rule by adding random items to its condition leads to a higher confidence (we will elaborate on this later in Section 3.4). So should we consider R_5 to be interesting or spurious? We will revisit this question after the introducing minimal predictive patterns.

3.3 PATTERN-BASED CLASSIFICATION

In the previous section, we discussed using pattern mining for finding rules that may help in knowledge discovery. In this section, we discuss using pattern mining for building classification models.

Earlier approaches in using patterns for classification focused on *associative classification*, which builds a rule-based classifier [Liu et al., 1998, Li et al., 2001b, Cong et al., 2005, Wang and Karypis, 2005, Veloso et al., 2006] (see Section 2.4.4).

Recently, the focus was more on using patterns to define *features* that can represent higher order interactions between the original data features [Cheng et al., 2007, Batal and Hauskrecht, 2010b]. The rationale behind this approach is that patterns (feature-value combinations) may capture more underlying semantics than simple features. Hence, the inclusion of some patterns can improve the classification performance.

Formally, given a dataset $D = \{x_i, y_i\}_{i=1}^{n}$ in d dimensional space and a set of patterns $\Omega = \{P_1, ..., P_m\}$, D is mapped into a higher dimensional space with $d + m$ dimensions by adding indicator binary features to the original features:

$$x_i \rightarrow x_i' = x_i \ \oplus \ \{b_{i,1}, ..., b_{i,m}\} \text{ where } b_{i,j} = 1 \text{ if } P_j \in x_i \text{ and } b_{i,j} = 0 \text{ if } P_j \notin x_i$$

The classification model is then learned in the new *expanded* feature space $D' = \{x_i', y_i\}_{i=1}^{n}$.

Example 5. *Consider the example in Figure 4, where there are two class labels y_1 and y_2 and the original data has three trinary features (F_1, F_2 and F_3). Assume we have the following two patterns: P_1: $F_1 = 1 \ \wedge \ F_3 = 2$ and P_2: $F_2 = 2$. Using this information, we can map the data into a higher dimensional space by defining two additional binary features b_1 and b_2, where b_1 (b_2) indicates the presence or absence of pattern P_1 (P_2) in each data instance. After performing this dimensionality expansion, it becomes very easy to classify the data (e.g., using a linear model).*

Note that applying frequent pattern mining usually returns a large number of frequent patterns, most of which may be irrelevant to the classification task (patterns are generated solely based on their support, not based on their discriminative power). Using all of these

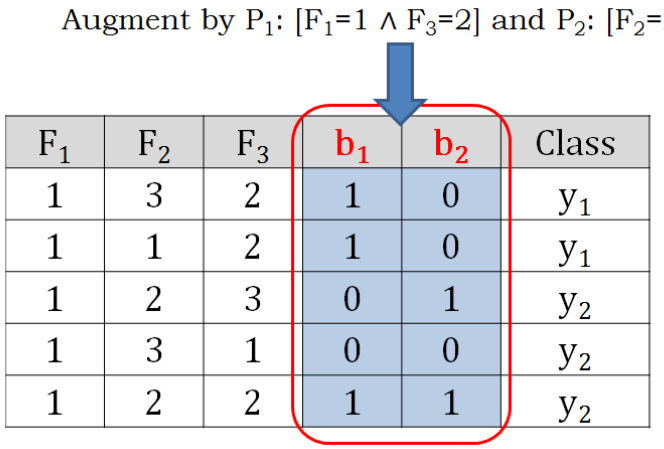


Figure 4: An example illustrating how to expand the original feature space (defined by F_1, F_2 and F_3) with features that correspond to more complex patterns. Binary features b_1 and b_2 correspond to patterns $F_1=1 \wedge F_3=2$ and $F_2=2$, respectively.

patterns as features may hurt the classification performance due to the curse of dimensionality. Therefore, it is important to select a subset of frequent patterns that are important for classification.

The most common approach for pattern-based classification is to evaluate each frequent pattern individually in order to select the most discriminative patterns [Nijssen et al., 2009, Cheng et al., 2007, Deshpande et al., 2005, Bay and Pazzani, 2001, Li et al., 2001b, Morishita and Sese, 2000]. However, as we discussed earlier, this approach usually leads to many spurious patterns in the results.

One way to partially overcome this problem is to apply an iterative forward feature selection method. In [Cheng et al., 2007], the authors defined a redundancy score (based on the Jaccard score [Geng and Hamilton, 2006]) and selected the classification patterns in an incremental way as follows: a pattern is added to the set of patterns if it is both *predictive* and has *low redundancy* to the patterns that are already selected. However, such iterative methods can be computationally expensive when applied on a large number of frequent patterns.

Having discussed these problems, it is important to develop a method that considers the relations among the patterns to ensure that the selected ones are highly *predictive* and at the same time contain *low redundancy*.

3.4 THE SPURIOUS PATTERNS PROBLEM

The task of selecting predictive patterns from a large pool of frequent patterns is more challenging than the standard task of feature selection. This is due to the *nested structure* of patterns: if a pattern is frequent, all of its subpatterns are also frequent, hence are also in the mining result. This nested structure causes the problem of *spurious patterns*, which we discuss and analyze in this section.

Definition 1. *A pattern P is a **spurious pattern** if P is predictive when evaluated by itself, but it is redundant given one of its subpatterns. Spurious patterns are formed by adding irrelevant items to other simpler predictive patterns.*

To illustrate this definition, consider the Bayesian belief network example in Figure 5. In this network, the value of the class variable Y only depends on the value of feature F_1 and is independent of the values of the other features: $Y \perp\!\!\!\perp F_i : i \in \{2,...,n\}$. Assume that pattern $P : F_1 = 1$ is highly predictive of class $Y = y_1$, so that $Pr(Y = y_1|P) > Pr(Y = y_1)$. Clearly, P is the only important pattern for predicting y_1.

Now consider a spurious pattern P' that is a *superpattern* of P, $P' : F_1 = 1 \wedge F_{q_1} = v_{q_1} \wedge ... \wedge F_{q_k} = v_{q_k}$, where $F_{q_i} \in \{F_2,...,F_n\}$ and v_{q_i} is *any possible value* of F_{q_i}. The network structure implies that $Pr(Y = y_1 | P') = Pr(Y = y_1 | P)$, hence $Pr(Y = y_1 | P')$ is also larger than the prior $Pr(Y = y_1)$.

The problem is that if we evaluate the patterns individually (without considering the nested structure of the patterns), we may falsely think that P' is a predictive pattern because the confidence of rule $P' \Rightarrow y_1$ is significantly larger than y_1 prior. However, P' is totally redundant given the real pattern P. Even by requiring a complex rule to have a higher confidence than their simplifications (the confidence improvement) [Bayardo, 1999, Grosskreutz

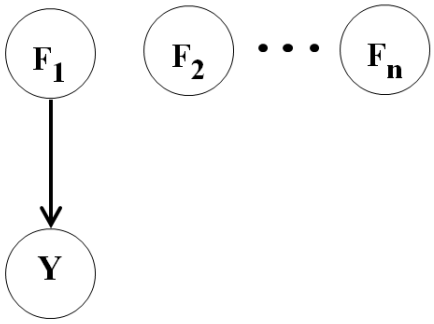


Figure 5: Illustrating the problem of spurious patterns in frequent pattern mining.

et al., 2010, Webb, 2007, Li et al., 2001b, Li et al., 2001a], the problem still exists and many spurious patterns can easily satisfy this constraint due to noise in sampling. Having spurious patterns in the mining results is undesirable for both *knowledge discovery* and *classification*. For knowledge discovery, spurious patterns can easily overwhelm the domain expert and prevent him/her from understanding the real causalities in the data. For classification, spurious patterns can lead to redundant and highly correlated features, which may negatively affect the classification performance. Therefore, it is very important to devise a method that can effectively filter out spurious patterns from the result.

3.5 MINING MINIMAL PREDICTIVE PATTERNS

In this section, we present our approach for mining minimal predictive patterns. We start by defining a Bayesian score to evaluate the predictiveness of a pattern compared to a more general population. After that, we introduce the concept of minimal predictive patterns to address the problem of spurious patterns. Lastly, we present an efficient mining algorithm that integrates pattern evaluation and frequent pattern mining.

3.5.1 Evaluating Patterns using the Bayesian Score

3.5.1.1 Classical Evaluation Measures A large number of measures have been proposed in the literature to evaluate the interestingness of individual rules. Examples of such measures include confidence, lift, weighted relative accuracy, J-measure, and others (cf [Geng and Hamilton, 2006]). Most of these measures trade-off two factors: 1) the *distributional unusualness* of the class variable in the population covered by the rule compared to the general population and 2) the *coverage* of the rule, which reflects its generality [Lavrač and Gamberger, 2005, Novak et al., 2009]. This trade-off is often achieved in an ad-hoc way, for instance by simply multiplying these two factors as in the weighted relative accuracy score [Kavsek and Lavrač, 2006] or in the J-measure [Smyth and Goodman, 1992]. Furthermore, most measures rely on point estimates of these quantities, often using the maximum likelihood estimation, and they do not capture the uncertainty of the estimation. In the following, we present a novel evaluation measure based on the Bayesian principal.

3.5.1.2 The Bayesian Score Suppose we want to evaluate how predictive is pattern P of class y compared to a more general group of instances G that contains the instances covered by P: $G_P \subseteq G$. We will denote our proposed Bayesian score for rule $P \Rightarrow y$ compared to group G by $\boldsymbol{BS(P \Rightarrow y, G)}$. We want this score to be high when there is a strong evidence in the data to support the hypothesis that the probability of y in the instances covered by P is higher than the probability of y in the instances of G that are not covered by P. The Bayesian score treats these probabilities as uncertain random variables as opposed to just relying on their point estimates as in the classical measures.

To explain the Bayesian score, we begin by defining the following three models:

1. M_e is the model that conjectures that all instances in group G have the ***same probability*** of having class y.
2. M_h is the model that conjectures that the probability of y in G_P is ***higher*** than the probability of y in the instances of G that are not covered by P ($G \setminus G_P$).
3. M_l is the model that conjectures that the probability of y in G_P is ***lower*** than the probability of y in the instances of G that are not covered by P.

The Bayesian score is based on the idea of scoring the marginal likelihoods of these three models. Intuitively, the more likely is model M_h compared to the other models, the more confident we are that pattern P is predictive of class y, hence the higher $BS(P \Rightarrow y, G)$ should be.

Let us start by defining the marginal likelihood for model M_e. This model assumes that all instances in G have the same probability of y, even though we are uncertain what that probability is. Let us denote $Pr(Y = y|G)$ by θ. To represent our uncertainty about θ, we use a beta distribution with parameters α and β. Let N_{*1} be the number of instances in G with class $Y = y$ and let N_{*2} be the number of instances in G with class $Y \neq y$. The marginal likelihood for model M_e is as follows:

$$Pr(G|M_e) = \int_{\theta=0}^{1} \theta^{N_{*1}} \cdot (1-\theta)^{N_{*2}} \cdot beta(\theta; \alpha, \beta) d\theta$$

The above integral yields the following well known closed-form solution [Heckerman et al., 1995]:

$$Pr(G|M_e) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha+N_{*1}+\beta+N_{*2})} \cdot \frac{\Gamma(\alpha+N_{*1})}{\Gamma(\alpha)} \cdot \frac{\Gamma(\beta+N_{*2})}{\Gamma(\beta)} \tag{3.1}$$

where Γ is the gamma function.

Now let us now define the marginal likelihood for model M_h. This model assumes that the probability of y in G_P, denoted by θ_1, is different from the probability of y in the instances of G that are not covered by P ($G \setminus G_P$), denoted by θ_2. Furthermore, M_h believes that θ_1 is higher than θ_2. To represent our uncertainty about θ_1, we use a beta distribution with parameters α_1 and β_1. To represent our uncertainty about θ_2, we use a beta distribution with parameters α_2 and β_2. Let N_{11} and N_{12} be the number of instances in G_P with class $Y = y$ and with class $Y \neq y$, respectively. That is, N_{11} is the number of **true positives** (instances correctly classified by the rule) and N_{12} is the number of **false positives** (instances incorrectly classified by the rule). Let N_{21} and N_{22} be the number of instances outside G_P with $Y = y$ and with $Y \neq y$, respectively (see Figure 6). Note that $N_{*1} = N_{11} + N_{21}$ and $N_{*2} = N_{12} + N_{22}$.

The marginal likelihood for model M_h is defined as follows:

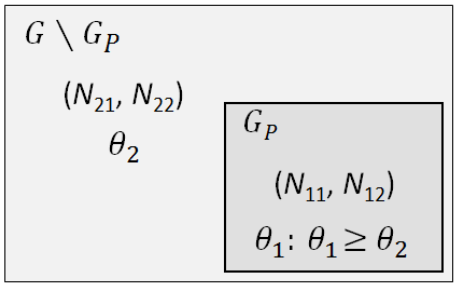


Figure 6: A diagram illustrating model M_h.

$$Pr(G|M_h) = \int_{\theta_1=0}^{1} \int_{\theta_2=0}^{\theta_1} {\theta_1}^{N_{11}} \cdot (1-\theta_1)^{N_{12}} \cdot {\theta_2}^{N_{21}} \cdot (1-\theta_2)^{N_{22}} \cdot \frac{beta(\theta_1;\alpha_1,\beta_1) \cdot beta(\theta_2;\alpha_2,\beta_2)}{k} d\theta_2 d\theta_1 \quad (3.2)$$

where k is a normalization constant for the parameter prior. Note that this formula does not assume that the parameters are independent, but rather constrains θ_1 to be higher than θ_2.

Below we show the closed-form solution we obtained by solving Equation 3.2. The derivation of this solution is fully described in the dissertation's appendix.

$$Pr(G|M_h) = \frac{1}{k} \cdot \frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\Gamma(\beta_1)} \cdot \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2)\Gamma(\beta_2)} \cdot \sum_{j=a}^{a+b-1} \left(\frac{\Gamma(a)\Gamma(b)}{\Gamma(j+1)\Gamma(a+b-j)} \cdot \frac{\Gamma(c+j)\Gamma(a+b+d-j-1)}{\Gamma(a+b+c+d-1)} \right) \quad (3.3)$$

where $a = N_{21} + \alpha_2$, $b = N_{22} + \beta_2$, $c = N_{11} + \alpha_1$, $d = N_{12} + \beta_1$. We solve for the normalization constant k by applying Equation 3.3 (without the k term) with $a = \alpha_2$, $b = \beta_2$, $c = \alpha_1$ and $d = \beta_1$. Note that $k = \frac{1}{2}$ if we use uniform priors on both parameters by setting $\alpha_1 = \beta_1 = \alpha_2 = \beta_2 = 1$.

Equation 3.3 can be expressed in logarithmic form to avoid computing very large numbers (to preserve numerical precision). Its computational complexity is $O(b) = O(N_{22} + \beta_2)$ (the number of terms in the summation). It turns out that we can redefine the solution of

Equation 3.2 so that its computational complexity is $O(min(N_{11}+\alpha_1, N_{12}+\beta_1, N_{21}+\alpha_2, N_{22}+\beta_2))$. The modifications that achieve this complexity result are also described in the appendix.

Lastly, let us define the marginal likelihood for model M_l, which assumes that θ_1 is lower than θ_2. The marginal likelihood for M_l is similar to Equation 3.2, but integrates θ_2 from 0 to 1 and constrains θ_1 to be integrated from 0 to θ_2 (forcing θ_1 to be smaller than θ_2). The solution for $P(G|M_l)$ can be computed with complexity $O(1)$ by reusing the terms computed in Equation 3.3, which is described in the appendix.

Now that we have computed the marginal likelihood of the three models: $Pr(G|M_e)$, $Pr(G|M_h)$ and $Pr(G|M_l)$, we compute the posterior probability of M_h (the model of interest) using Bayes theorem:

$$Pr(M_h|G) = \frac{Pr(G|M_h)Pr(M_h)}{Pr(G|M_e)Pr(M_e)+Pr(G|M_h)Pr(M_h)+Pr(G|M_l)Pr(M_l)} \tag{3.4}$$

To be "non-informative", we might simply assume that all three models are a priori equally likely: $Pr(M_e) = Pr(M_h) = Pr(M_l) = \frac{1}{3}$.

Note that Equation 3.4 quantifies in a Bayesian way how a posteriori likely is model M_h. Since this is the quantity we are interested in, $Pr(M_h|G)$ is used to score rule $P \Rightarrow y$ compared to group G: $BS(P \Rightarrow y, G) = Pr(M_h|G)$.

Example 6. *Let us use the Bayesian score to evaluate rule R_3: Family history = Yes ⇒ CHD in Example 4. This rule covers 30 CHD cases ($N_{11} = 30$) and 20 controls ($N_{12} = 20$). Let us compute $BS(R_3, G_\phi)$ for evaluating the predictiveness of R_3 compared to the entire dataset. Using the notations introduced earlier, $N_{*1} = 50$ and $N_{*2} = 100$ (the number of CHD cases and controls in the dataset). Also, $N_{21} = N_{*1} - N_{11} = 20$ and $N_{22} = N_{*2} - N_{12} = 80$. Let us use uniform beta priors for all parameters: $\alpha = \beta = \alpha_1 = \beta_1 = \alpha_2 = \beta_2 = 1$. The likelihood of M_e is 3.2×10^{-43}, the likelihood of M_h is 1.5×10^{-38} and the likelihood of M_l is 1×10^{-44}. Using Equation 3.4, we get $BS(R_3, G_\phi) = Pr(M_h|G_\phi) = 0.99998$. This implies that there is a strong evidence in the data to conclude that pattern Family history=yes makes CHD more likely.*

Example 7. *Figure 7 shows a plot that illustrates the Bayesian score. In this plot, we set* $N_{*1} = N_{*2} = 30$*. We vary each of* N_{11} *and* N_{12} *from 0 to 30 and plot the logit (log-odds) of the corresponding Bayesian score*[1]*. We can see that the score is an increasing function of* N_{11} *(the number of true positives) and a decreasing function of* N_{12} *(the number of false positives). It achieves its maximum when* $N_{11} = N_{*1}$ *and* $N_{12} = 0$ *(the pattern correctly classifies all the instances).*

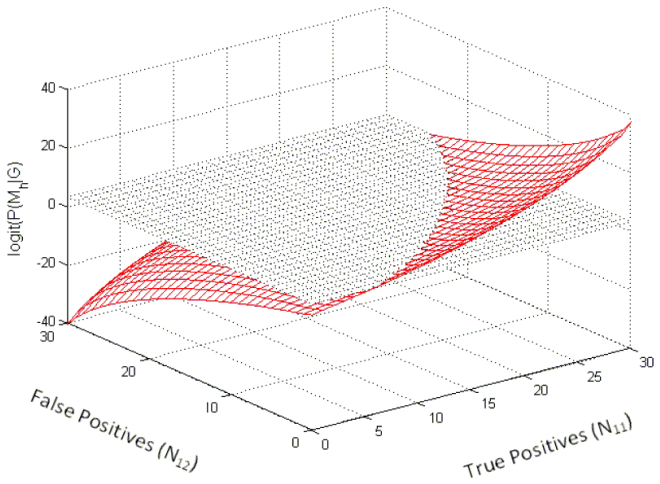


Figure 7: A plot showing the logit function of the Bayesian score as a function of the true positives and the false positives. Points above the plane have a Bayesian score larger than 0.95.

3.5.2 Minimal Predictive Patterns

The Bayesian score proposed in the previous section provides a way to evaluate the predictiveness of a pattern by contrasting it to a more general population than the population covered by the pattern. A straightforward approach for mining predictive patterns is to apply the Bayesian score to compare each pattern to the entire data and report the top scoring patterns to the user. However, this approach does not overcome the spurious patterns prob-

[1]The logit function of a probability p (the inverse of the logistic function) is monotonically increasing in p. We plot the logit of the score instead of the actual score because it has a smoother transition, making it easier to visualize.

lem we discussed in Section 3.4: If rule $P \Rightarrow y$ has a very high Bayesian score (pattern P is highly predictive of class y), many rules $P' \Rightarrow y$ that correspond to spurious patterns $P' \supset P$ are expected to have a high Bayesian score as well (provided that P' have enough support in the data). As a result, the rules presented to the user would contain a lot of redundancies and fail to provide a good coverage of the data.

Now we introduce the minimal predictive patterns framework for selecting patterns that are highly predictive of the class variable and at the same time contain low redundancy.

Definition 2. *A pattern P is a **Minimal Predictive Pattern (MPP)** with respect to class label y if P predicts y significantly better than **all** of its subpatterns:*

$$\forall S \subset P \ : \ BS(P \Rightarrow y, G_S) \geq \delta$$

Where δ is a user specified significance parameter.

This definition means that if P is an MPP with respect to class y, then there is a strong evidence in the data not only to conclude that P improves the prediction of y compared to the entire data, but also compared to the data matching any of its subpatterns. Notice that the MPP definition prefers simple patterns over more complex patterns (the Occam's Razor principal) because a pattern is not an MPP if its effect on the class distribution "can be explained" by a simpler pattern (subpattern) that covers a larger population.

We would like to note that detecting MPP patterns can also be done, in addition to using our Bayesian score, using a frequentist statistical significance test. We used this approach in our earlier work [Batal and Hauskrecht, 2010a]. A similar method based on Fisher exact test has also been used by [Wong et al., 2005] for disease outbreak detection on biosurveillance data.

Example 8. *Let us go back to Example 4. We want to decide whether rule R_5: Race = Caucasian $\wedge$ Family history = Yes $\Rightarrow$ CHD is an interesting rule (we should show it to the user) or a spurious rule (we should remove it from the results). This rule covers 21 CHD cases and 11 controls. In order to be an MPP, pattern Race = Caucasian $\wedge$ Family history = Yes should predict CHD significantly better than all of its subpatterns. If we compare R_5 to the entire dataset, we get $BS(R_5, G_\phi) = 0.9997$. If we compare R_5 to its subrule R_2, we get $BS(R_5, G_{R_2}) = 0.9998$. Finally, if we compare R_5 to its subrule R_3, we get $BS(R_5, G_{R_3}) = 0.47$.*

We can see that R_5 is considered very predictive when compared to the entire dataset or to subrule R_2, but is not predictive when compared to subrule R_3. Therefore, we do not consider R_5 an important rule because it is equivocal whether it predicts CHD as being more likely than does R_3.

Example 9. *Let us consider again the simple Bayesian network in Figure 5 to illustrate how the MPP framework tackles the problem of spurious patterns. Assume we have 10 binary features (F_1 to F_{10}) and a binary class variable: $dom(Y) = \{y_1, y_2\}$. Assume the CPTs of the network are defined as follows: $Pr(F_i = 1) = 0.4 : i \in \{1, ..., 10\}$, $Pr(Y = y_1 | F_1 = 1) = 0.9$ and $Pr(Y = y_1 | F_1 = 0) = 0.5$. Let the data D be 500 instances that are randomly generated from this network. Our task is to mine patterns that are predictive of class y_1[2]. As we discussed earlier, the only important rule in D for predicting y_1 is $F_1 = 1 \Rightarrow y_1$. All other rules are spurious.*

Let us use frequent pattern mining to explore patterns that occur in more than 10% of the data. Doing so, we obtain 1,257 frequent patterns (potential rules). If we apply the χ^2 test with significance level $\alpha = 0.05$ to select rules that have a significant positive correlation with class y_1, we get 284 rules that are statistically significant. Even if we apply the False Discovery Rate (FDR) technique [Benjamini and Hochberg, 1995] to correct for multiple hypothesis testing, we get 245 significant rules! Note that these methods do not overcome the spurious patterns problem. Let us now apply the confidence improvement constraint to filter out "non-productive" rules [Bayardo, 1999, Grosskreutz et al., 2010, Webb, 2007, Li et al., 2001b, Li et al., 2001a]. By doing so, we get 451 rules! This clearly demonstrates that the confidence improvement constraint is an ineffective criterion. Lastly, let us mine rules that correspond to MPPs for y_1 using significance parameter $\delta = 0.95$. Doing so, we obtain only a single rule $F_1 = 1 \Rightarrow y_1$ (the only important rule) and effectively filter out all other spurious rules[3].

[2] The prior of y_1 in this network is $Pr(Y = y_1) = 0.66$.

[3] The 0.95 significant parameter is chosen so that it is comparable to the commonly used frequentist 0.05 significance level.

3.5.3 The Mining Algorithm

In this section, we present the algorithm for mining the minimal predictive patterns. The algorithm utilizes frequent pattern mining to explore the space of potential patterns (rules) and applies the Bayesian score to select the ones that satisfy the MPP definition (Definition 2).

To search the space of patterns, we partition the data according to the class labels and explore frequent patterns *for each class* $y \in dom(Y)$ *separately* using a *local minimum support* σ_y that is related to the number of instances from class y. This approach is reasonable when pattern mining is applied in the supervised setting for the following reasons:

1. For unbalanced data, mining frequent patterns using a global minimum support may result in missing many important patterns in the rare classes.
2. Mining patterns that are frequent in one of the classes (hence potentially predictive for that class) is more efficient than mining patterns that are globally frequent[4].

Figure 8 shows an example of a dataset with two class labels y_1 and y_2. The data is partitioned into two parts based on the class label and MPPs are mined from each partition using a local minimum support that is related to the partition size. Finally, the class specific MPPs are combined to form the final result.

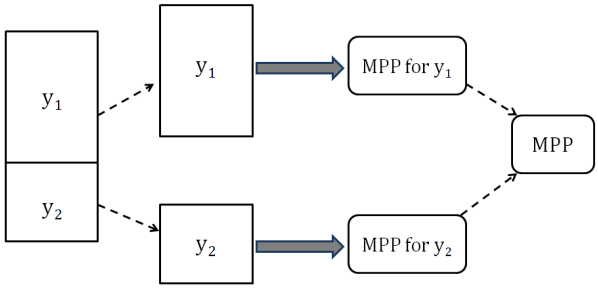


Figure 8: The Mining is applied on each class label separately and then final result is obtained by combining the class specific MPPs.

[4]It is much more efficient to mine patterns that cover more than n instances in one of the classes as opposed to mining all patterns that cover more than n instances in the entire database (the former is always a subset of the latter).

The algorithm for mining MPPs for class label $y \in dom(Y)$ takes as input the following arguments:

1. The data instances from class y: $D_y = \{(x_i, y_i) : y_i = y\}$.
2. The data instances that do not belong to class y: $D_{\neg y} = \{(x_i, y_i) : y_i \neq y\}$.
3. The local minimum support threshold σ_y.
4. The significance parameter δ.

The algorithm explores the space of frequent patterns and outputs patterns (rules) that satisfy the MPP definition.

A straightforward way to obtain the result is to apply the commonly used *two-phase* approach as in [Cheng et al., 2007, Webb, 2007, Xin et al., 2006, Kavsek and Lavrač, 2006, Exarchos et al., 2008, Deshpande et al., 2005, Li et al., 2001b], which generates all frequent patterns in the first phase and evaluates them in the second phase (a post-processing phase). That is, this approach would apply the following two phases to mine MPPs for class y:

1. Phase I: Mine *all* frequent patterns: $FP = \{P_1, ..., P_m : sup(P_i, D_y) \geq \sigma_y\}$.
2. Phase II: For each pattern $P_i \in FP$, output rule $P_i \Rightarrow y$ if $BS(P_i \Rightarrow y, G_S) \geq \delta \ : \ \forall S \subset P_i$.

In contrast to this two-phase approach, our algorithm *integrates pattern evaluation with frequent pattern mining*. This allows us to apply additional pruning techniques that are not applicable in the two-phase approach.

Our algorithm explores the lattice of frequent patterns level by level from the bottom-up. It starts by exploring frequent *1-patterns*, then frequent *2-patterns*, and so on. Whenever the algorithm visits a frequent pattern P (a node in the lattice), it computes its Bayesian score with respect to its subpattens and adds it to result if it satisfies the MPP definition.

Example 10. *Assume that our data D contains 100 instances from class y_1 and 100 instances from class y_2 and that our task is to find MPPs for class y_1. Figure 9 illustrates the mining algorithm on a small frequent pattern lattice (with three items I_1, I_2 and I_3). Every frequent k-pattern is represented as a node in the lattice with k children: one child for each of its (k-1)-subpatterns. Next to each pattern P, we show the number of instances from class y_1 that satisfy P ($\#y_1$), the number of instances from class y_2 that satisfy P ($\#y_2$) and the confidence of rule $P \Rightarrow y_1$ ($conf = \frac{\#y_1}{(\#y_1 + \#y_2)}$).*

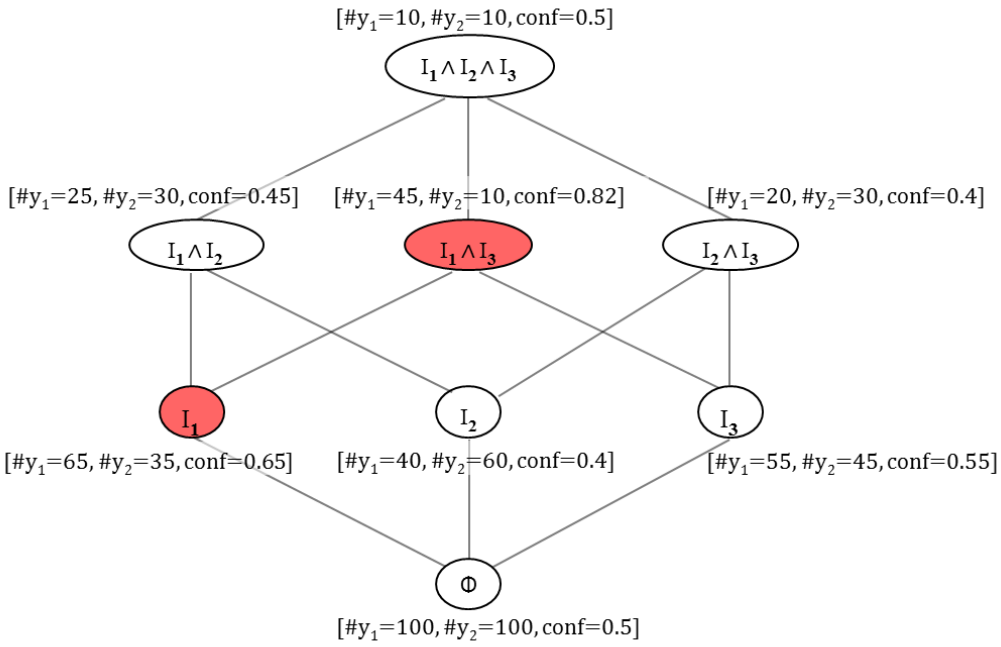


Figure 9: An illustrative example showing the frequent pattern lattice associated with $I_1 \wedge I_2 \wedge I_3$. Next to each pattern (node), we show the number of instances from class y_1 and y_2 that the pattern covers. We also show the confidence of the rule that predicts y_1. The MPPs for class y_1 are shown in red.

The algorithm works by exploring the frequent pattern lattice level by level. It first explores the first level to find MPPs of size 1. Assuming a significance parameter $\delta = 0.95$*, pattern* I_1 *is an MPP because* $BS(I_1 \Rightarrow y_1, G_\phi) \geq \delta$*. Next, the algorithm explore the second level of the lattice to find MPPs of size 2. Pattern* $I_1 \wedge I_3$ *is an MPP because* $BS(I_1 \wedge I_3 \Rightarrow y_1, G_\phi) \geq \delta$*,* $BS(I_1 \wedge I_3 \Rightarrow y_1, G_{I_1}) \geq \delta$ *and* $BS(I_1 \wedge I_3 \Rightarrow y_1, G_{I_3}) \geq \delta$*. After that, the algorithm explores the third level and so on.*

3.5.4 Pruning the Search Space

In this section, we illustrate how integrating MPP evaluation with frequent pattern mining helps pruning the search space (speeding up the mining). We say that pattern P is **pruned** if we *do not explore any of its superpatterns*. This can be seen as *excluding the entire sublattice with bottom P from the lattice of patterns*.

Frequent pattern mining relies only on the *support* information to prune patterns according to the Apriori property: Any *infrequent* pattern is pruned because its superpatterns are guaranteed not to be frequent (see Section 2.2.1).

However, frequent pattern mining can be computationally very expensive, especially when:

- The data dimensionality is high.
- The minimum support is low.
- The data features are highly correlated.

These reasons cause the frequent pattern lattice to become extremely large. One simple way to speed up the mining is to raise the minimum support threshold. However, doing so may result in missing many important predictive patterns. In fact, [Cheng et al., 2007] argued that the predictive power of very high support patterns is often limited[5].

In the following, we present two effective pruning techniques that can utilize the *predictiveness* of patterns to further prune the search space. The first technique is *lossless* and the second is *lossy*.

3.5.4.1 Lossless pruning The MPP definition can help us to prune the search space. The idea is to prune pattern P if we guarantee that none of its superpatterns is going to be an MPP. However, since patterns are explored in a level-wise fashion, we do not know the class distribution in the superpatterns of P. But we know that

$$\forall P' \supset P \ : \ G_{P'} \subseteq G_P \implies sup(P', D_y) \leq sup(P, D_y) \ \wedge \ sup(P', D_{\neg y}) \leq sup(P, D_{\neg y})$$

We now define the **optimal superpattern** of P with respect to class y, denoted as P^*, to be

[5] This is analogous to the uselessness of stop words for document classification or retrieval.

a hypothetical pattern that covers all instances from y and none of the instances from the other classes:

$$sup(P^*, D_y) = sup(P, D_y) \ \wedge \ sup(P^*, D_{\neg y}) = 0$$

P^* is the best possible superpattern for predicting y that P can generate. Now, we *safely* prune P if P^* does not satisfy the MPP definition. That is, if P^* does not predict y significantly better than P or one of its subpatterns:

$$\text{Prune } P \text{ if } \exists\, S \subseteq P : \ BS(P^* \Rightarrow y, G_S) < \delta$$

Note that this pruning technique does not miss any MPP (lossless) and it is an *anti-monotone* test in the sense that if a pattern fails to pass the test, all of its superpatterns will fail the same test as well (see Section 2.3.3).

Example 11. *Consider pattern $P = I_1 \wedge I_2 \wedge I_3$ in Figure 9. This pattern covers 10 instances from y_1 and 10 instances from y_2. P cannot generate any MPP for y_1 because even its optimal superpattern P^* is not significant compared to subpattern $I_1 \wedge I_3$: $BS(P^* \Rightarrow y_1, G_{I_1 \wedge I_3}) = 0.62 < \delta = 0.95$. Therefore, we can safely prune P because we know that none of its superpatterns will be an MPP for y_1.*

3.5.4.2 Lossy pruning This technique performs *lossy* pruning, which means that it speeds up the mining, but at the risk of missing some MPPs. The idea is that for mining MPPs for class y, we prune pattern P if the underlying probability of y in the instances covered by P is lower than the probability of y in the entire data: $Pr(Y = y|G_P) < Pr(Y = y|G_\phi)$. To decide whether this is the case, we apply our Bayesian score to evaluate rule $P \Rightarrow y$ compared to G_ϕ and we prune P if model M_l (the model that assumes the probability of y in P is lower than outside P) is the most likely model (see Section 3.5.1.2).

Let us now explain the rationale behind this heuristic. Assume we are interested in mining MPPs for class y and we have pattern P where the probability of y in G_P is lower than the probability of y in the data. To give a concrete example, let us say that we have a census income data and we are interested in MPPs for the class *Income* = *High* (e.g., people who make over 80K a year). Let us say that the probability of *Income* = *High* is low for

people that match pattern P : *Education* = *High-School* (people with high school degree as their highest eduction level).

For every pattern P' that is a superpattern of P, we can write the following:

$$P' \supset P \implies \exists\, P'' :\, P' = P \,\wedge\, P'' \implies G_{P'} = G_P \,\cap\, G_{P''}$$

Going back to our running example, P' can be *Education* = *High school* $\wedge$ *Gender* = *Female*. The population of instances covered by P' is the intersection of the population covered by P : *Education* = *High school* and the population covered by P'' : *Gender* = *Female* as shown in Figure 10.

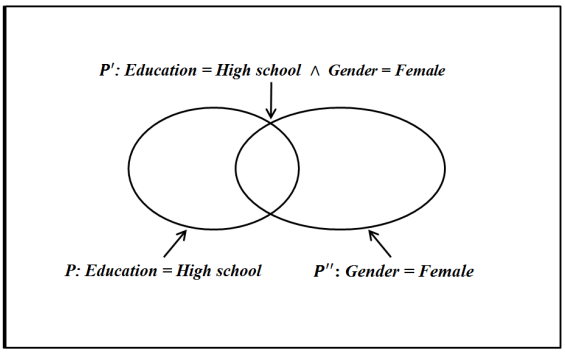


Figure 10: If the probability of class *Income* = *High* in the population defined by *Education* = *High school* is low, we do not expect the probability of *Income* = *High* in the population defined by *Education* = *High school* $\wedge$ *Gender* = *Female* to be significantly higher than the probability of *Income* = *High* in the population defined by *Gender* = *Female*.

Now in order for P' to be an MPP, it should predict y significantly better than all of its subpatterns, including P''. However, we know that the probability of y in G_P is low, hence it is very unlikely that the probability of y in $G_P \,\cap\, G_{P''}$ will be significantly higher than the probability of y in $G_{P''}$.

Example 12. *Consider pattern I_2 in Figure 9. This pattern covers 40 instances from y_1 and 60 instances from y_2. If we apply the Bayesian score to evaluate rule $I_2 \Rightarrow y_1$ compared to G_ϕ, we find that M_l is the most likely model. Therefore, we prune I_2 because it is unlikely to generate any MPP for y_1. That is, we do not generate its superpatterns $I_1 \wedge I_2$, $I_2 \wedge I_3$ and $I_1 \wedge I_2 \wedge I_3$.*

3.6 EXPERIMENTAL EVALUATION

3.6.1 UCI Datasets

For our experimental evaluation, we use 15 publicly available datasets from the UCI machine learning repository [Asuncion and Newman, 2007]. Recall that pattern mining methods require the data to be discrete (see Section 2.1). We discretize numerical attributes using [Fayyad and Irani, 1993] supervised discretization method, which recursively applies entropy minimization and uses the minimum description length criterion to decide on the number of bins.

Table 5 shows the main characteristics of these datasets. For each dataset, we show the number of instances, the number of attributes, the number of items (distinct attribute-value pairs after discretization) and the number of classes.

3.6.2 Quality of Top-K Rules

In this section, we present our experiments for supervised descriptive rule discovery (see Section 3.2). The purpose is to show that the MPP framework is able to explain and cover the data with fewer rules than existing methods, which is beneficial for knowledge discovery. We present the results first on a synthetic dataset and then on the UCI datasets.

3.6.2.1 Compared Methods We compare the quality of the top rules for the following rule evaluation measures:

Dataset	# Instances	# Attributes	# Items	# Classes
Lymphography	142	18	57	2
Parkinson	195	22	51	2
Heart	270	13	33	2
Hepatitis	155	19	39	2
Diabetes	768	8	19	2
Breast cancer	286	9	41	2
Nursery	12,630	8	27	3
Red wine	1,599	11	32	3
Mammographic	961	5	13	2
Tic tac toe	958	9	27	2
Ionosphere	351	34	145	2
Kr vs kp	3,196	36	73	2
Pen digits	10,992	16	141	10
Zoo	74	16	32	3
WDBC	569	30	94	2

Table 5: UCI datasets characteristics.

1. ***GR:*** Rules are ranked using the Growth Rate measure, which was used in [Dong and Li, 1999, Bailey et al., 2002, Yu et al., 2012] in the context of *emerging pattern mining*.

$$GR(P \Rightarrow y) = \frac{sup(P, D_y)}{sup(P, D_{\neg y})} \cdot \frac{|D_{\neg y}|}{|D_y|}$$

where D_y and $D_{\neg y}$ represent the instances from class y and not from class y, respectively.

2. ***J-measure:*** Rules are ranked using the J-measure [Smyth and Goodman, 1992], a popular information theoretic measure that scores the rules by their information content.

$$J\text{-}measure(P \Rightarrow y) = \frac{sup(P,D)}{|D|} \times \sum_{z \in \{y, \neg y\}} conf(P \Rightarrow z) \cdot log_2 \left(\frac{conf(P \Rightarrow z)}{conf(\Phi \Rightarrow z)} \right)$$

3. ***WRAcc:*** Rules are ranked using the Weighted Relative Accuracy, which was used in [Kavsek and Lavrač, 2006] in the context of *subgroup discovery*[6].

$$WRAcc(P \Rightarrow y) = \frac{sup(P,D)}{|D|} \times (conf(P \Rightarrow y) - conf(\Phi \Rightarrow y))$$

   Note that this measure is compatible (provides the same rule ranking) with the support difference measure used in [Bay and Pazzani, 2001] for *contrast set mining* (see [Novak et al., 2009] for more details).

4. ***BS:*** Rules are ranked using our proposed Bayesian score. However, this method scores each rule individually with respect to the entire data and do not consider the relations between the rules.

5. ***Conf-imp:*** Only rules that satisfy the confidence improvement constraint are retained [Bayardo, 1999, Grosskreutz et al., 2010, Webb, 2007, Li et al., 2001b, Li et al., 2001a] and they are ranked according to their confidence.

6. ***MPP:*** Our proposed method, which mines rules that correspond to MPPs (patterns that satisfy Definition 2 using significance parameter $\delta = 0.95$) and rank them according to the Bayesian score.

Note that the *GR* measure does not consider the coverage of the rule when assessing its interestingness. For example, *GR* favors a rule that covers 8% of the instances of in one class and 1% of the instances in the other classes over a rule that covers 70% of the instances of in one class and 10% of the instances in the other classes (because $\frac{8}{1} > \frac{70}{10}$). As a result, *GR* often chooses rules that are very specific (with low coverage) and do not generalize well.

To overcome this, the *J-measure* and *WRAcc* explicitly incorporate the rule coverage $\frac{Sup(P,D)}{|D|}$ in their evaluation functions to favor high coverage rules over low coverage rules. This is done by multiplying the coverage with a factor that quantifies the distributional

[6]The algorithm by [Kavsek and Lavrač, 2006] uses weighted sequential covering and modifies the WRAcc measure to handel example weights.

surprise (unusualness) of the class variable in the rule (the cross entropy for *J-measure* and the relative accuracy for *WRAcc*). However, it is not clear whether simply multiplying these two factors leads to the optimal trade-off. On the other hand, our Bayesian score (used by *BS* and *MPP*) achieves this trade-off implicitly by properly modeling the uncertainty of the estimation (the more data we have, the more certain is our estimation).

Note that the first four methods (*GR*, *J-measure*, *WRAcc* and *BS*) evaluate the rules individually with respect to the entire data without considering their relations. On the other hand, *Conf-imp* and *MPP* evaluate each rule with respect to all of its subrules. *Conf-imp* simply requires each rule have a higher confidence than its subrules, while *MPP* requires each rule to be significantly more predictive than its subrules according to Definition 2.

For all methods, we use frequent pattern mining to explore the space of potential rules and we set the local minimum support (σ_y) to 10% the number of instance in the class. For *BS* and *MPP*, we use uniform beta priors (uninformative priors) for all parameters when computing the Bayesian score. We apply both the *lossless pruning* 3.5.4.1 and the *lossy pruning* 3.5.4.2 for the *MPP* method.

3.6.2.2 Performance Measures We evaluate the quality of the top rules that are induced by the different evaluation measures in terms of their classification performance and their representation in the ROC space.

Classification performance: For a set of rules to be practically useful, the rules should accurately predict the class label of unseen data instances (*high precision*) and the rule set should provide a good coverage of the data (*high recall*).

We compare the different evaluation measures according to the classification performance of the top rules. In particular, for each of the compared measures, we mine top k rules from the training data and use them to classify the testing data. The classification is done according to the highest confidence rule in the set of top rules [Liu et al., 1998]:

$$Prediction(x) = \arg\max_{y_i} \{\, conf(P \Rightarrow y_i) \colon P \in x \,\}$$

We evaluated the classification performance using the *F1 score* [Sebastiani, 2002], which is the harmonic mean of the precision and recall. We compute it using micro-averaging,

which constructs a global contingency table to calculate precision and recall. The reason we use micro-average F1 score instead of the classification accuracy is because it differentiates between the error of an *unclassified* instance (not covered by any rule) and the error of a *misclassified* instance (incorrectly classified). This is because an unclassified instance only negatively affects the recall, while a misclassified instance negatively affects both precision and recall. Note that if all test instances are classified (covered) by the rule set, then micro-average F1 score is the same as the classification accuracy.

ROC space representation: Another way to evaluate the quality of rules is to analyze them in the ROC (Receiver Operating Characteristics) space, as suggested by [Lavrač and Gamberger, 2005, Kavsek and Lavrač, 2006, Novak et al., 2009].

The ROC space [Fawcett, 2006] is a two-dimensional space that shows classifier performance in terms of its *False Positive Rate* (*FPR*) (the fraction of false positives out of the negatives) plotted on the X-axis, and *True Positive Rate* (*TPR*) (the fraction of true positives out of the positives) plotted on the Y-axis.

To analyze rules in the ROC space, we find the top rules that predict a specific class label and represent each rule by its *FPR* and *TPR*. Rules that lie close to the main diagonal are usually considered as insignificant because the distribution of positives and negatives in the population they cover is similar to the distribution in the entire dataset. According to [Lavrač and Gamberger, 2005, Kavsek and Lavrač, 2006, Novak et al., 2009], the most significant rules correspond to the points from which the ROC convex hull is constructed. Consequently, the area under the ROC convex hull (AUC) was used to measure the combined quality of the rule set. Figure 11 shows a set of 5 rules in the ROC space and their convex hull, which is defined by R_1, R_3 and R_4.

All reported results (for both classification performance and area under the ROC convex hull) are obtained using *10-fold cross-validation*, where the same train/test splits are applied for all compared methods.

3.6.2.3 Results on Synthetic Data We first start by showing results on a synthetic dataset (generated from pre-defined patterns). This dataset is obtained by randomly sampling 500 instances from the Bayesian network in Figure 12. In this network, we have 20

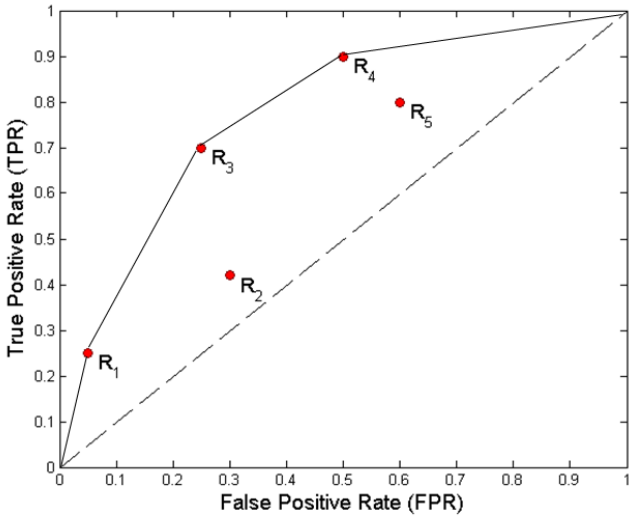


Figure 11: Representation of 5 rules in the ROC space and their convex hull.

binary features (F_1 to F_{20}) and a binary class variable Y: $dom(Y) = \{y_1, y_2\}$. The value of Y only depends on the values of features F_1 and F_2 and is independent of the values of the other features: $Y \perp\!\!\!\perp F_i$ for $i \in \{3,...,20\}$. Features F_1 and F_2 follow a bernoulli distribution with probability 0.5 and features F_3 to F_{20} follow a bernoulli distribution with probability 0.25. Note that there are only four predictive rules in this data: $F_1 = 1 \wedge F_2 = 1 \Rightarrow y_1$, $F_1 = 1 \wedge F_2 = 0 \Rightarrow y_2$, $F_1 = 0 \wedge F_2 = 1 \Rightarrow y_2$ and $F_1 = 0 \wedge F_2 = 0 \Rightarrow y_1$ with confidences 0.95, 0.95, 0.8 and 0.8, respectively. All other rules are spurious.

Figure 13 shows the classification performance for the different rule evaluation measures. The X-axis shows the number of top rules that are used for classification and the Y-axis shows their F1 score.

We can see that MPP achieves the optimal classification performance using only the top four rules, which correspond to the true predictive rules in this data. In comparison, the other methods require much more rules to achieve the same performance because they favor many spurious rules (such as $F_1 = 1 \wedge F_2 = 1 \wedge F_5 = 0 \Rightarrow y_1$) over more important rules (such as $F_1 = 0 \wedge F_2 = 0 \Rightarrow y_1$). As a result, the top rules contain a lot of redundancies

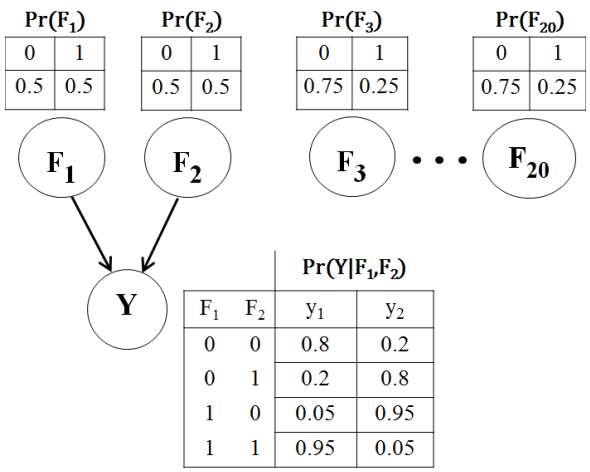


Figure 12: The Bayesian belief network used to generate the synthetic dataset.

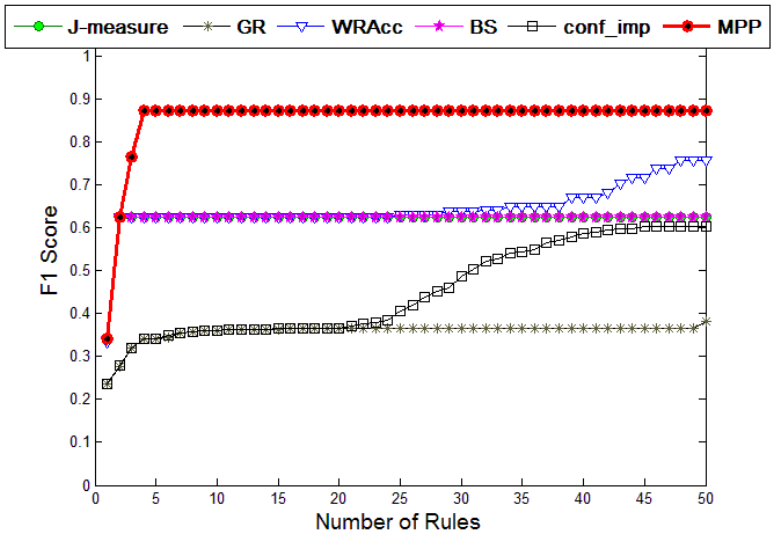


Figure 13: Comparing the performance of the different rule evaluation measures (see Section 3.6.2.1) on the synthetic dataset. The X-axis is the number of the top rules and the Y-axis is the F1 score of the rule set.

and fail to cover the patterns of this data.

Let us now discuss the ROC space representation. In Figure 14, we plot rules R_1 :

$F_1 = 1 \wedge F_2 = 1 \Rightarrow y_1$ and $R_2 : F_1 = 0 \wedge F_2 = 0 \Rightarrow y_1$, which we know are both important for predicting class y_1. According to the ROC representation, R_2 is considered suboptimal because it lies below the ROC convex hull. This means that a rule mining method that finds only rule R_1 has the same "quality" (measured using the area under the ROC convex hull) as a method that finds both R_1 and R_2. However, this is not the case because we need both R_1 and R_2 to classify the instances of y_1 (notice that the population cover by R_1 is disjoint from the population covered by R_2).

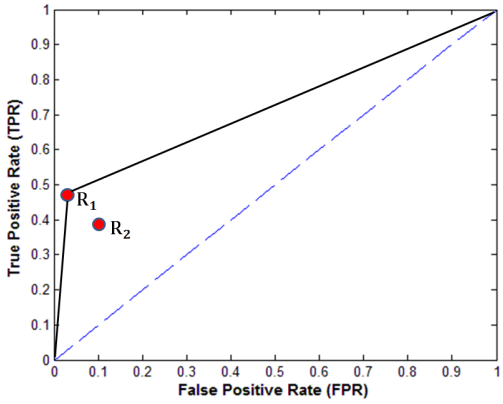


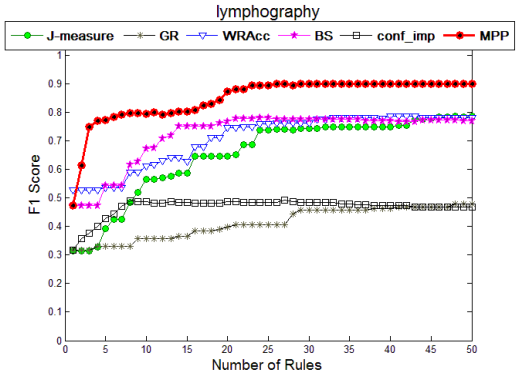
Figure 14: Illustrating the deficiency of the ROC space representation on the synthetic data. The figure shows rule $R_1 : F_1 = 1 \wedge F_2 = 1 \Rightarrow y_1$ and rule $R_2 : F_1 = 0 \wedge F_2 = 0 \Rightarrow y_1$. Note that R_2 is not on the convex hull.

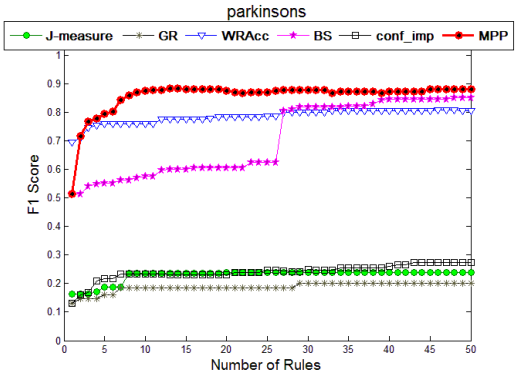
3.6.2.4 Results on UCI Datasets Figure 15 shows the classification performance for the different evaluation measures on the UCI datasets. We can see that *GR* is the worst performing method for most datasets. The reason is that rules with the highest *GR* scores are usually very specific (have low coverage) that may easily overfit the training data. The other measures (*J-measure*, *WRAcc* and *BS*) perform better than *GR* because they favor high-coverage rules over low-coverage rules, which results in rules that generalize better on the testing data. However, because these measures do not consider the relations among the rules, the top scoring rules usually contain many spurious rules. As a result, they fail to provide a good coverage of the data (see for example the *lymphography* and the *zoo* datasets). Finally, we can see that for most datasets, *MPP* achieves the best performance

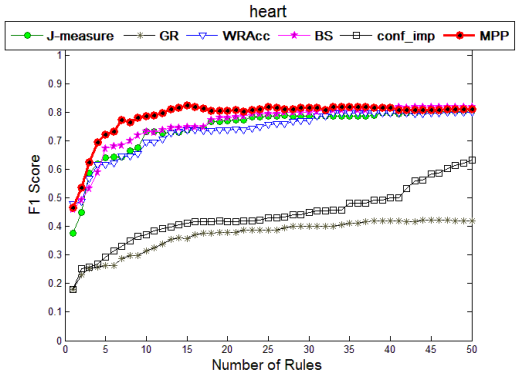
with the smallest number of rules.

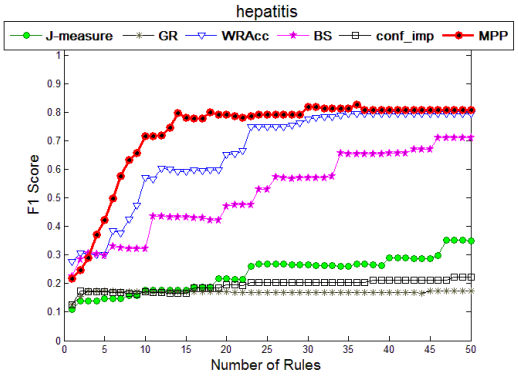
Table 6 shows the area under the rules' ROC convex hull for the different rule evaluation measures on the UCI datasets. For each measure, we mine the *top 5 rules* from each class label, compute their AUC and finally average all class-specific AUCs[7]. The results are reported using averages obtained via *10-fold cross-validation*. We can see that *MPP* is the best performing method on *eleven* out of the fifteen datasets.

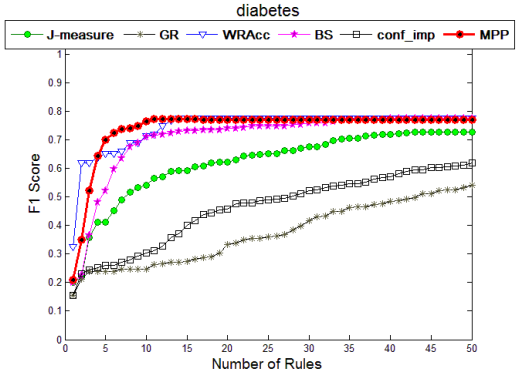
[7]For each class label $y \in dom(Y)$, we mine the top 5 rules for predicting y from the training data and compute their AUC on the testing data. Then we average the class specific AUCs.

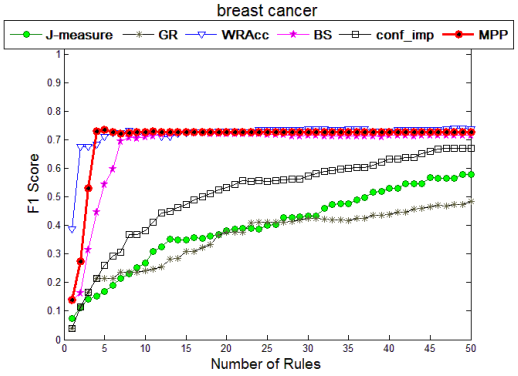

lymphography
J-measure
GR
WRAcc
BS
conf_imp
MPP
F1 Score
Number of Rules

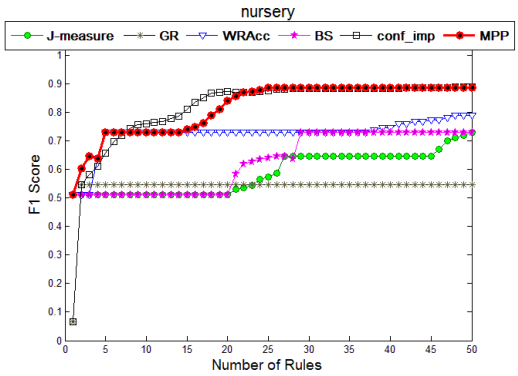

parkinsons
J-measure
GR
WRAcc
BS
conf_imp
MPP
F1 Score
Number of Rules

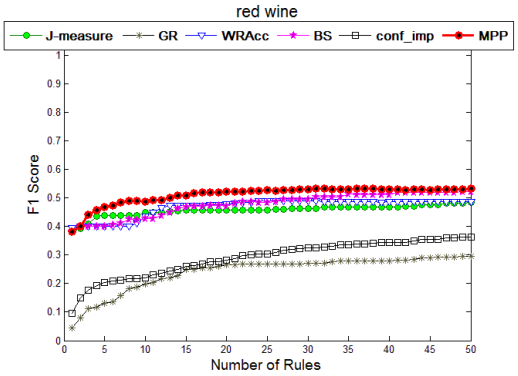

heart
J-measure
GR
WRAcc
BS
conf_imp
MPP
F1 Score
Number of Rules


hepatitis
J-measure
GR
WRAcc
BS
conf_imp
MPP
F1 Score
Number of Rules


diabetes
J-measure
GR
WRAcc
BS
conf_imp
MPP
F1 Score
Number of Rules


breast cancer
J-measure
GR
WRAcc
BS
conf_imp
MPP
F1 Score
Number of Rules


nursery
J-measure
GR
WRAcc
BS
conf_imp
MPP
F1 Score
Number of Rules


red wine
J-measure
GR
WRAcc
BS
conf_imp
MPP
F1 Score
Number of Rules

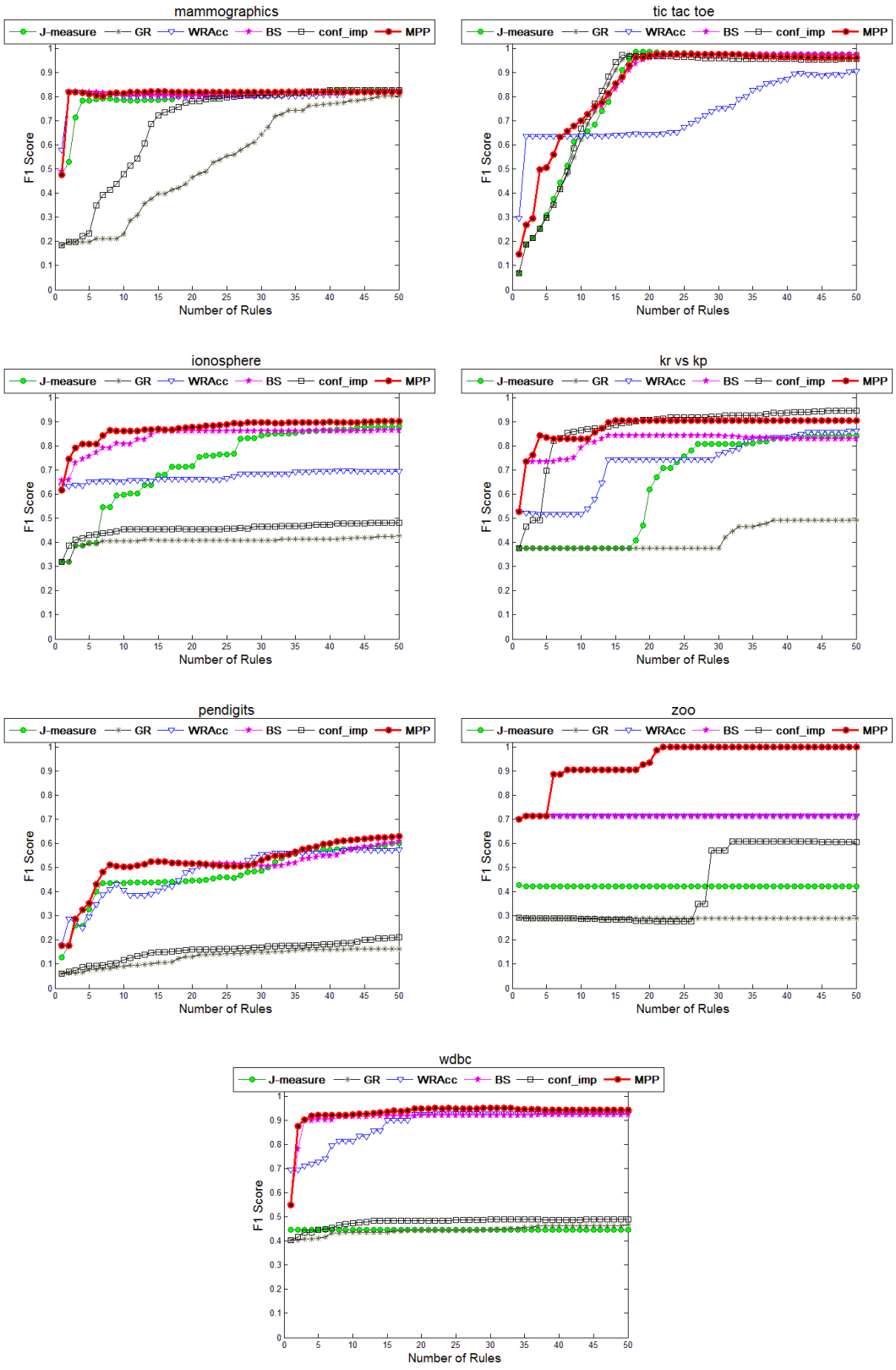


Figure 15: Comparing the performance of the different rule evaluation measures (see Section 3.6.2.1) on the UCI datasets. The X-axis is the number of the top rules and the Y-axis is the F1 score of the rule set.

Dataset	*GR*	*J-measure*	*WRAcc*	*BS*	*conf-imp*	*MPP*
Lymphography	71.5	76.68	79.01	77.14	74.4	**85.43**
Parkinson	70.61	73.31	76.4	76.22	75.9	**84.32**
Heart	63.08	76.44	**81.51**	79.76	63.82	80.13
Hepatitis	72.01	70.97	83.47	83.92	74.65	**85.51**
Diabetes	58.48	65.66	72.66	69.42	58.57	**73.64**
Breast cancer	57.57	60.01	69.76	68.62	59.21	**70.29**
Nursery	72.83	**80.94**	80.76	**80.94**	72.88	**80.94**
Red wine	56.72	65.16	68.58	67.35	57.28	**68.9**
Mammographic	62.05	81.26	85.29	**85.54**	63.75	85.41
Tic tac toe	59.77	59.77	68.1	**68.8**	59.77	**68.8**
Ionosphere	75.44	78.81	81.82	80.82	76.71	**82.02**
Kr vs kp	72.15	78.87	79.04	78.62	72.21	**84.17**
Pen digits	61.41	80.64	**83.08**	81.67	61.49	81.9
Zoo	100	100	100	100	100	100
WDBC	81.85	90.3	90.64	90.53	82.6	**90.89**
# wins	0	1	2	3	0	11

Table 6: The area under the rules' ROC convex hull for the different rule evaluation measures (see Section 3.6.2.1) on the UCI datasets.

3.6.3 Pattern-based Classification

In this section, we present our experiments for classification. The purpose is to show that mapping the data into a higher dimensional space using MPPs as additional features (see

Section 3.3) can boost the classification performance. We present the results first on a synthetic dataset and then on the UCI datasets in Section 3.6.1.

3.6.3.1 Compared Methods

We compare the performance of the following classifiers:

1. ***SVM:*** The linear Support Vector Machine (SVM) classifier [Vapnik, 1995] on the original feature space. We optimize the cost parameter (a parameter that controls regularization) using internal cross-validation.
2. ***SVM RBF:*** SVM with the Radial Basis Function (RBF) kernel (Gaussian kernel). We optimize the cost parameter and the kernel width parameter by a grid search using internal cross-validation.
3. ***DT:*** Decision tree using the CART (Classification And Regression Tree) [Breiman et al., 1984] algorithm.
4. ***Boost:*** The boosting algorithm with decision stumps (one level decision trees) as weak learners. We use the Gentle AdaBoost algorithm [Friedman et al., 2000] because it is known to be more robust than the original AdaBoost algorithm [Schapire and Singer, 1999]. We set the maximum number of boosting iterations to 100.
5. ***KNN:*** The k-nearest neighbors classifier. We set the number of neighbors $k = 5$ and use the Euclidean distance to measure the similarity between instances.
6. ***MBST:*** The recently proposed Model Based Search Tree (MBST) algorithm [Fan et al., 2008], which uses frequent pattern mining to build a decision tree. The basic idea is to partition the data in a top down manner. That is, at each node of the tree: 1) invoke a frequent pattern mining algorithm, 2) select the most discriminative pattern (according to IG), 3) divide the data into two subsets, one containing this pattern and the other not, and 4) repeat the process recursively on the two subsets.
7. ***FP-IG:*** This method ranks the frequent patterns according to information gain and selects the top **50** patterns for classification.
8. ***MPP:*** Our proposed method, which ranks the MPPs (patterns that satisfy Definition 2 using significance parameter $\delta = 0.95$) according to the Bayesian score and selects the top **50** patterns for classification.

The first five methods (*SVM*, *SVM RBF*, *DT*, *Boost* and *KNN*) are standard classification methods that work directly on the original feature space[8]. On the other hand, *MBST*, *FP-IG* and *MPP* are pattern based classification methods that expands the feature space with additional patterns. The classification model for *FP-IG* and *MPP* is built by learning a **linear SVM** classifier in the space of the original features plus the induced patterns (see Section 3.3).

For *FP-IG* and *MPP*, we set the local minimum support (σ_y) to 10% of the number of instances in the class. For *MBST*, we found that its performance (execution time and classification accuracy) is very sensitive to the invocation minimum support parameter. Unfortunately, the authors did not provide guidance on how to set this parameter. We tried many values and found that setting the invocation minimum support to 25% is a reasonable choice for our datasets. Note that setting it to lower values made *MBST* computationally prohibitive on several datasets.

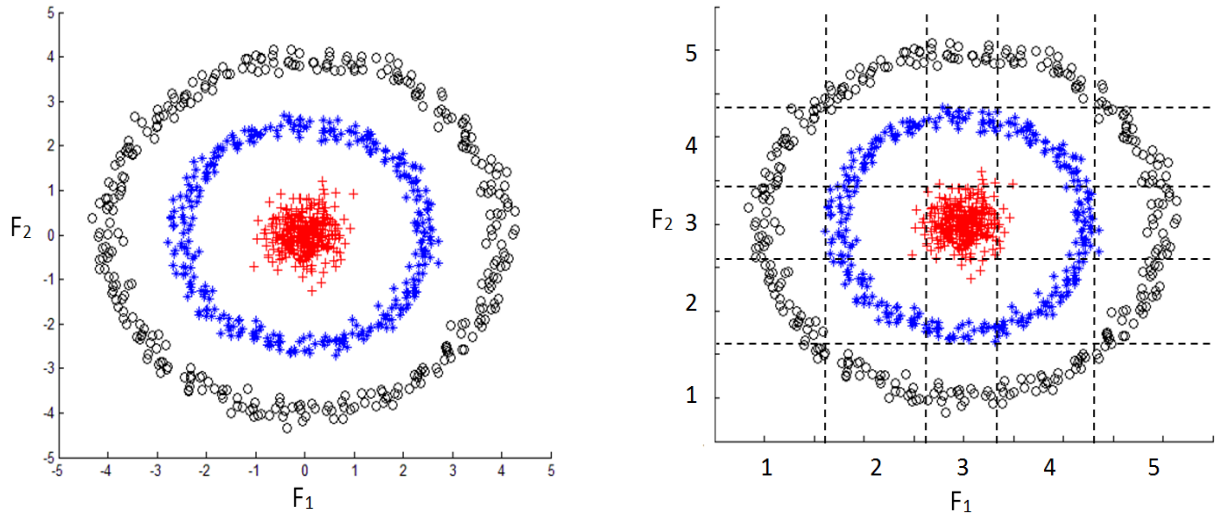
All results are reported using averages obtained via *10-fold cross-validation*, where the same train/test splits are used for all compared methods.

3.6.3.2 Results on Synthetic Data We first start by showing results on a synthetic dataset because this allows us to better understand the relation between the methods and their classification performance.

We use the ***3 circles*** dataset, which consists of 20 *numerical* features (F_1 to F_{20}) and a trinary class variable Y: $dom(Y) = \{y_1, y_2, y_3\}$. The relation between the class labels and features F_1 and F_2 is shown in Figure 16:a, where each class label is shown with a different color. The other features (F_3 to F_{20}) are just noisy features that follow a Gaussian distribution $N(0, \sigma^2)$, where σ^2 is approximately the same as the variance of F_1 and F_2.

Since this data is numeric, discretization should be applied in order to do pattern mining (for *MBST*, *FP-IG* and *MPP*). Figure 16:b shows the data after applying [Fayyad and Irani, 1993] supervised discretization method. As we see, F_1 and F_2 were divided into 5 bins each. The other noisy features were divided into 2 bins each (the minimum number of bins)

[8] *SVM RBF* implicitly maps the data into a high dimensional space, but performs all computations in the original feature space using the kernel trick



(a) (b)

Figure 16: On the left (a), we show the *3 circles* synthetic dataset projected on the 2D plane defined by features F_1 and F_2. On the right (b), we show the data after applying [Fayyad and Irani, 1993] discretization.

because they do not contain any discriminative signal.

Figure 17 shows the classification accuracy on the *3 circles* dataset. Let us first discuss the performance of the *classical classification* methods. Linear SVM completely fails on this datasets (with accuracy close to random guessing) because the classes cannot be linearly separated. *SVM RBF* improves the performance over linear SVM because of its ability to define non-linear decision boundaries. However, because the data contain a lot of noise (only 2 out of the 20 features are useful for classification), *SVM RBF* does not always generalize well to the testing examples. Both *DT* and *boost* perform well on this data. *KNN* performs poorly because the presence of noisy features makes the Euclidean distance is inappropriate for measuring similarity between the data instances.

Now, Let us discuss the performance of the *pattern-based classification* methods. Notice that the discretized *3 circles* dataset (Figure 16:b) contains several patterns that are important for classification. For example, all instances that satisfy pattern $F_1 = 1$ belong to the

black class and all instances that satisfy pattern $F_1 = 3 \ \wedge \ F_2 = 3$ belong to the *red* class.

Figure 17 shows that *MBST* also does not perform very well on this data and is outperformed by simple decision tree (*DT*). *FP-IG* performs poorly because it evaluates the patterns individually, hence it ends up selecting many spurious patterns that are not useful for classification. For example, a spurious pattern such as $F_1 = 3 \wedge \ \wedge \ F_2 = 3 \ \wedge \ F_{10} = 1$ have a higher information gain score than an important pattern such as $F_1 = 2$. Finally, *MPP* is the best performing method and achieves an accuracy of 98.9%.

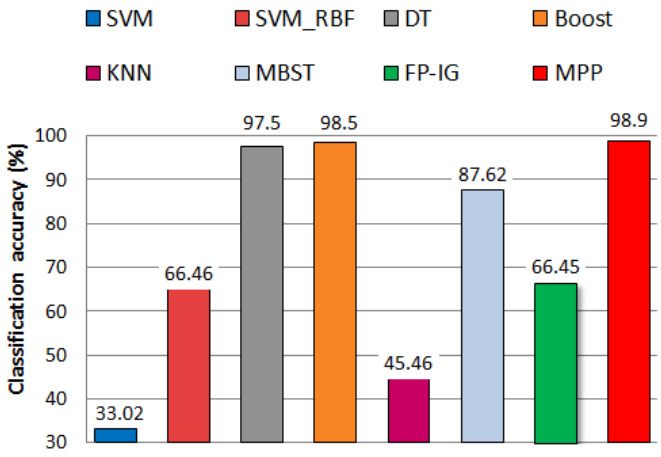


Figure 17: The classification accuracy (%) of the different classifiers (see Section 5.7.2.1) on the *3 circles* synthetic dataset.

3.6.3.3 Results on UCI Datasets Table 7 shows the classification accuracy on the UCI datasets. We can see that *MPP* is the best performing method on *seven* out of the fifteen datasets. Figure 18 summarizes the results of Table 7 in a graphical form.

Dataset	*SVM*	*SVM RBF*	*DT*	*Boost*	*KNN*	*MBST*	*FP-IG*	*MPP*
Lymphography	86.52	88	76.67	84.62	85.14	83.05	88.71	**90.05**
Parkinson	87.54	88.13	86.04	90.62	91.26	87.49	85.99	**92.10**
Heart	**85.19**	84.07	76.67	80	81.11	75.56	83.59	**85.19**
Hepatitis	78.86	85.19	72.23	79.17	80.62	83.17	82.42	**85.32**
Diabetes	75.91	77.08	71.10	76.04	74.10	**77.47**	76.17	76.43
Breast cancer	71.31	**75.15**	68.15	71.27	74.82	69.19	72.36	74.82
Nursery	75.61	84.08	87.59	75.74	78.14	88.69	89.32	**96.64**
Red wine	61.29	**69.10**	66.30	64.36	61.85	61.36	63.30	63.67
Mammographic	81.06	83.56	80.34	82.10	80.96	**83.76**	82.83	83.14
Tic tac toe	65.34	89.66	86.74	83.71	84.76	79.12	**100**	**100**
Ionosphere	86.04	**94.28**	88.56	93.45	84.65	87.17	92	92.30
Kr vs kp	95.99	97.94	**99.31**	96.56	94.68	95.77	96.81	97.03
Pen digits	98.29	**99.50**	95.78	96.89	99.32	68.52	98.81	98.74
Zoo	**100**	**100**	98.57	**100**	97.32	98.57	**100**	**100**
WDBC	97.72	96.66	92.45	**97.89**	96.83	94.22	97.72	97.72
# wins	2	5	1	2	0	2	2	7

Table 7: The classification accuracy (%) of the different classifiers (see Section 5.7.2.1) on the UCI datasets.

3.6.4 Mining Efficiency

In this section, we present our experiments for comparing the efficiency of several pattern mining methods. The purpose is to show that our proposed pruning techniques can greatly

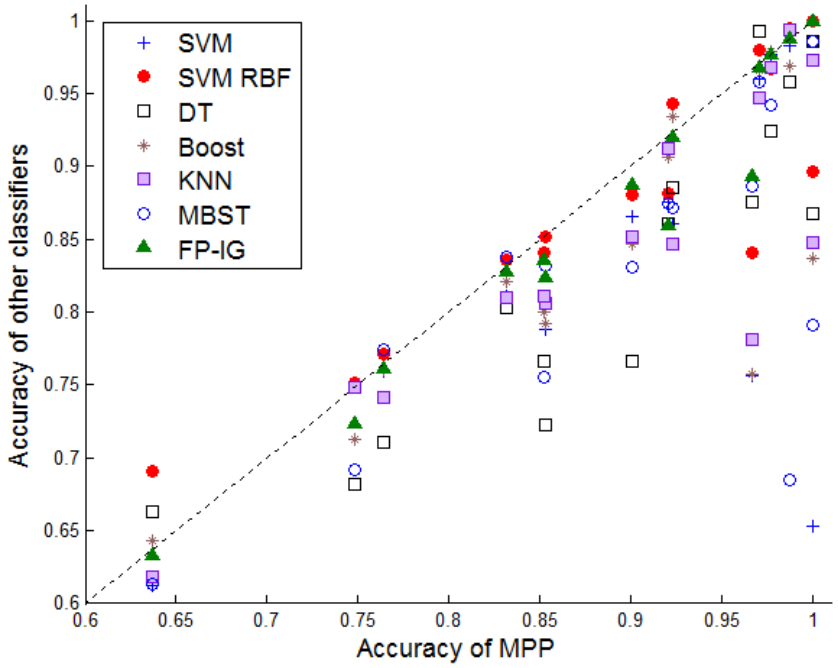


Figure 18: Comparing the classification accuracies for different methods on the UCI datasets against *MPP*. Every point corresponds to a specific method on a specific dataset: its y-coordinate represents its accuracy and its x-coordinate represents the accuracy of *MPP* for the same dataset. Points above the diagonal correspond to methods that outperform *MPP* and points below the diagonal correspond to methods that are outperformed by *MPP*.

improve the efficiency of frequent pattern mining.

3.6.4.1 Compared Methods

We compare the running time of the following methods:

1. ***FPM:*** A frequent pattern mining method, which corresponds to the first phase of any two-phase method [Cheng et al., 2007, Webb, 2007, Xin et al., 2006, Kavsek and Lavrač, 2006, Exarchos et al., 2008, Deshpande et al., 2005, Li et al., 2001b]. We use the algorithm by [Zaki, 2000], which applies the vertical data format (see Section 2.2.3).
2. ***MBST:*** The Model Based Search Tree (MBST) method [Fan et al., 2008], which uses frequent pattern mining to build a decision tree (see Section 5.7.2.1).

3. ***MPP-naïve:*** This method mines MPPs using a naive two-phase implementation, which first applies *FPM* (the first phase) and then evaluate all frequent patterns (the second phase).
4. ***MPP-lossless:*** This method mines MPPs by integrating pattern evaluation with frequent pattern mining. It applies *only the lossless pruning* described in Section 3.5.4.1.
5. ***MPP-lossy:*** Our proposed method, which mines MPPs by integrating pattern evaluation with frequent pattern mining. It applies *both the lossless pruning and the lossy pruning* described in Section 3.5.4.2.

The experiments are conducted on a Dell Precision T1600 machine with an Intel Xeon 3GHz CPU and 16GB of RAM. All methods are implemented in MATLAB.

3.6.4.2 Results on UCI Datasets Table 8 shows the execution time (in seconds) of the compared methods on the UCI datasets. We use the same settings as before (see Section 5.7.2.1): For *FPM*, *MPP-naïve*, *MPP-lossless* and *MPP-lossy*, we set the local minimum support (σ_y) to 10%. For *MBST*, we set the invocation minimum support to 25%.

The results show that scalability is a concern for *FPM*, *MPP-naïve* and *MBST*. For example, these methods are very slow on the *parkinson*, *hepatitis*, *ionoshpere* and *WDBC* datasets. In comparison, *MPP-lossless* is faster due to its lossless pruning and *MPP-lossy* is much faster due to its lossy pruning. Consider for instance the *parkinson* dataset. Mining all frequent patterns took 9,866 seconds, *MBST* took 5,159 seconds and *MPP-naïve* took 10,832 seconds. On the other hand, *MPP-lossless* took 828 seconds (an order of magnitude faster than *FPM*) and *MPP-lossy* took only 37 seconds (more than two orders of magnitude faster than *FPM*). This shows that our pruning techniques can significantly improve the mining efficiency.

Effectiveness of the lossy pruning: Remember that the lossy pruning heuristic used by *MPP-lossy* does not guarantee the completeness of the result (see Section 3.5.4.2). Here, we discuss its effectiveness by examining the number of MPPs that are missed when we apply the lossy pruning. In turned out that on *ten* out of the fifteen datasets (namely, *lymphography*, *heart*, *hepatitis*, *diabetes*, *breast cancer*, *nursery*, *mammographic*, *tic tac toe*, *ionosphere* and *zoo*), *MPP-lossy* did not miss any MPP. On the *parkinson*, *red wine*, *Kr vs*

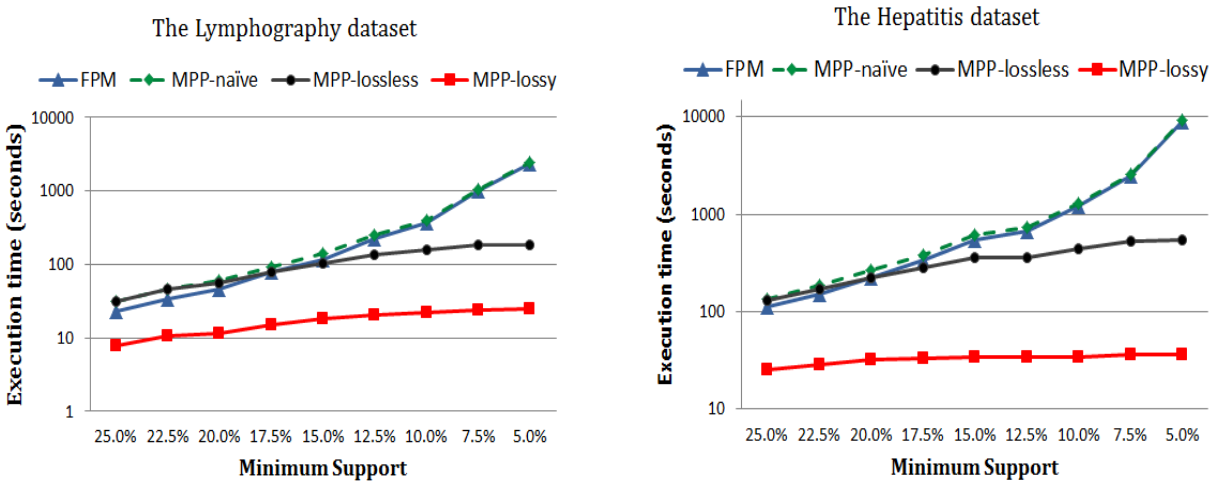
Dataset	*FPM*	*MBST*	*MPP-naïve*	*MPP-lossless*	*MPP-lossy*
Lymphography	335	568	344	154	22
Parkinson	9,866	5,159	10,832	828	37
Heart	41	143	57	38	9
Hepatitis	1,121	1,808	1,156	394	35
Diabetes	3	16	6	6	2
Breast cancer	3	13	4	4	2
Nursery	2	3	11	9	8
Red wine	24	73	52	52	11
Mammographic	1	2	1	1	1
Tic tac toe	2	7	4	4	3
Ionosphere	16,580	3,522	16,809	1,080	814
Kr vs kp	175	538	535	479	104
Pen digits	70	23	137	135	121
Zoo	183	116	242	23	4
WDBC	2,305	1,834	5,182	270	73

Table 8: The mining time (in seconds) for the different pattern mining methods (see Section 3.6.4.1) on the UCI datasets.

KP, *pen digits* and *WDBC* datasets, *MPP-lossy* misses on average 1.6%, 2.1%, 1.5%, 2.1% and 0.9% (respectively) of the total number of MPPs. This small loss in completeness can be often tolerated in exchange for the large gain in efficiency that is obtained by using the lossy pruning.

Changing the minimum support threshold: Let us now compare the execution time

of the different methods using different minimum support thresholds. Figure 19 shows the execution time (on logarithmic scale) of *FPM*, *MPP-naïve*, *MPP-lossless* and *MPP-lossy* on the *lymphography* dataset and on the *hepatitis* dataset. We did not include *MBST* because it is very inefficient for low minimum support thresholds.



(a) (b)

Figure 19: The mining time (on logarithmic scale) for the different pattern mining methods using different support thresholds on the *lymphography* dataset (a) and the *hepatitis* dataset (b).

We can see that the execution time of *FPM* and *MPP-naïve* exponentially blows up when the minimum support decreases. On the other hand, *MPP-lossless* controls the complexity and the execution time increases much slower when the minimum support decreases. For example, by setting the minimum support to 5% for *hepatitis*, *MPP-lossless* becomes more than 15 times faster than *FPM*. Finally, *MPP-lossy* is very fast and it scales up much better than the other methods when the minimum support is low.

3.7 SUMMARY

In this chapter, we studied pattern mining in the supervised setting and presented the minimal predictive patterns (MPP) framework. Our framework relies on Bayesian inference to evaluate the predictiveness of patterns. It also considers the structure of the patterns to ensure that each pattern in the result offers a significant predictive advantage over all of its generalizations. We presented an efficient algorithm for mining the MPP set. In contrast to the widely used two-phase approach, our algorithm integrates pattern evaluation with frequent pattern mining and applies several pruning techniques to improve the efficiency.

Our experimental evaluation on several synthetic and real-world data illustrates the following benefits of our work:

1. The MPP framework is able to explain and cover the data with fewer rules than existing rule mining methods, which facilitates the process of knowledge discovery.
2. Pattern-based classification using MPPs outperforms many well known classifiers.
3. Mining MPPs is more efficient than mining all frequent patterns.

4.0 TEMPORAL PATTERN MINING

In Chapter 2, we discussed mainly the related research on pattern mining for attribute-value data (atemporal data). Now, we focus our attention to temporal data, which require different tools and techniques than those used for atemporal data.

This chapter provides an overview of the related research on temporal pattern mining. Our objective is not just to enumerate the methods proposed so far, but also to classify and organize them in a way that makes it easier to compare and contrast the different methods.

The rest of this chapter is organized as follows. Section 4.1 provides our categorization of the different types of temporal data, which will be used throughout the chapter. Section 4.2 reviews methods for classifying temporal data. Section 4.3 describes pattern mining methods for time point data. Section 4.4 describes pattern mining methods for time interval data. Section 4.5 describes temporal abstraction techniques for converting numeric time series into time interval sequences. Finally, Section 4.6 summarizes the chapter.

4.1 TEMPORAL DATA MODELS

Temporal data usually refers to any type of data that explicitly or implicitly capture the notion of time or define a specific order. For example, even when time is not explicit and only an ordering is given, we may still consider the data to be temporal (e.g., DNA sequences).

We say that the temporal data is **univariate** if the data instances consist of measurements of a single variable over time. We say that the data is **multivariate** if the data instances consist of measurements of multiple variables over time. Multivariate temporal data appear in a wide variety of domains, such as health care [Hauskrecht et al., 2010, Sac-

chi et al., 2007, Ho et al., 2003], sensor measurements [Jain et al., 2004], intrusion detection [Lee et al., 2000], motion capture [Li et al., 2009], environmental monitoring [Papadimitriou et al., 2005] and many more.

We say that the temporal data is **regularly sampled in time** if the time between consecutive events is uniform (the same for all pairs of consecutive events). Otherwise, the data is **irregularly sampled in time**. The latter is often the case for electronic health records[1], which is the focus of this thesis.

Temporal data can be also classified based on the values of its observations. If the values are numerical, we have **numeric time series** (see Figure 20:a). If the values are discrete (belong to a finite alphabet Σ), we have **symbolic sequences**. For example, a DNA sequence (Figure 20:b) is a symbolic sequence, where the alphabet represents the 4 possible nucleotides $\Sigma = \{A, G, C, T\}$. Figure 20:c shows an example of a *multivariate symbolic sequence* that is *irregularly sampled in time*. In this example, there are 3 temporal variables F_1, F_2 and F_3 and the observations belong to alphabet $\Sigma = \{A, B, C, D\}$[2]. A real world example of such multivariate symbolic sequences are log messages that are emitted from multiple machines or alarms that are emitted in a telecommunication network [Mannila et al., 1997]. Note that symbolic sequences can also be obtained from numeric time series using discretization [Lin et al., 2003, Yang et al., 2005].

In some cases, the data do not consist of time points, but of *time intervals* that have durations and are associated with specific start and end times. For example, the data may express temporal concepts such as: "the patient underwent cancer chemotherapy from day 11 until day 15 of his hospitalization". In this case, we have **state sequences**, where each state holds during a specific time interval. Figure 20:d shows an example of a *multivariate state sequence*, where there are 3 temporal variables F_1, F_2 and F_3 and the states belong to alphabet $\Sigma = \{A, B, C, D\}$. Note that state sequences can also be obtained from numeric time series using temporal abstraction, which will be discussed in Section 4.5.

Finally, for each temporal data model, the database may consist of a **single long se-**

[1] The time period between the consecutive lab measurements usually varies within the same patient (depending on his clinical situation) and also across different patients.

[2] If the different temporal variables take different sets of values, the alphabet is defined as the union of all such values.

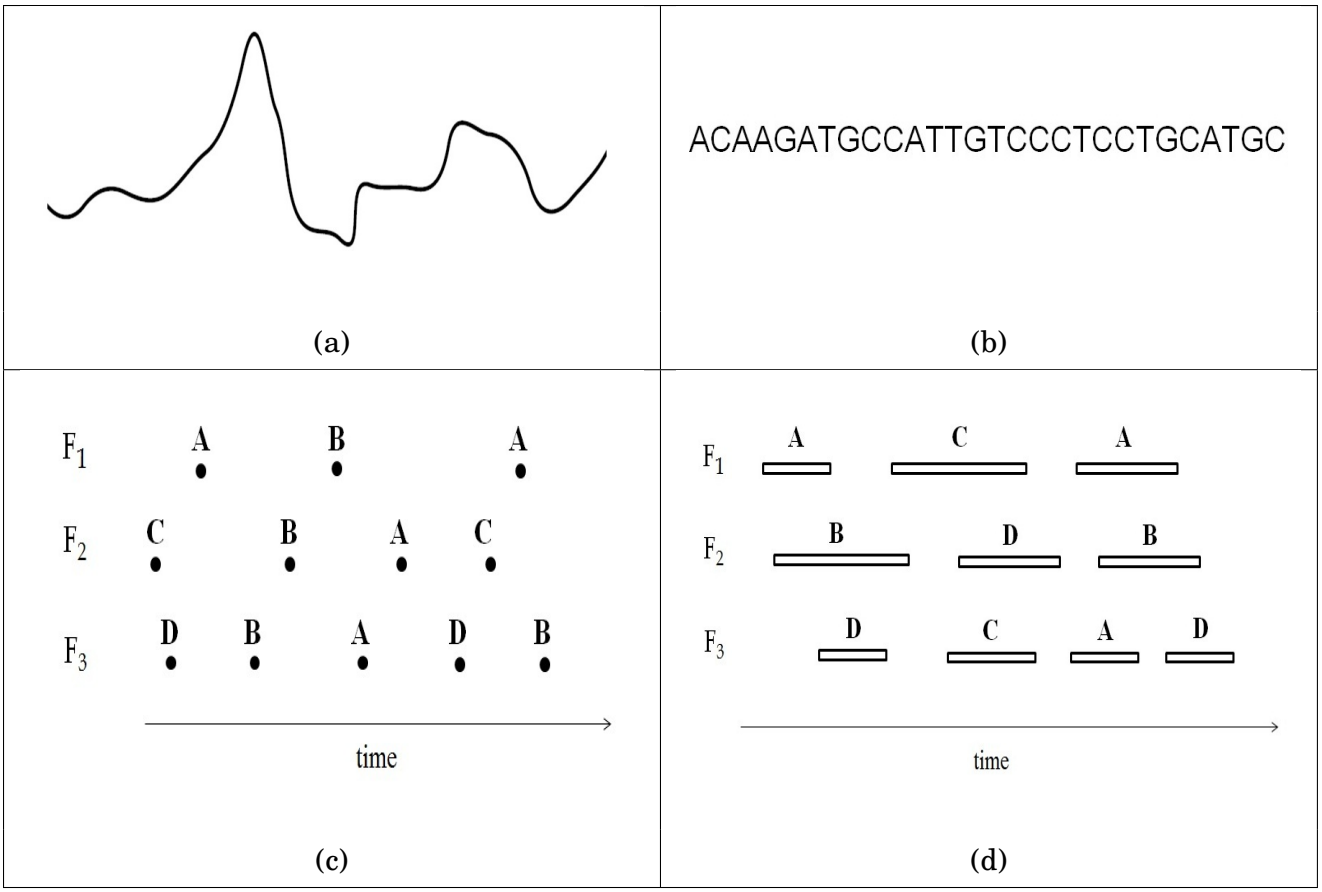


Figure 20: Illustrating different temporal data models: the upper left (a) is a univariate numeric time series, the upper right (b) is a univariate symbolic sequence, the bottom left (c) is a multivariate symbolic sequence and the bottom right (d) is a multivariate state sequence.

quence or **multiple (short) sequences**. Examples of the former are weather data [Höppner, 2003] or stock market data (may be collected over several years). Examples of the latter are web-click data, customer shopping profiles [Agrawal and Srikant, 1995], telephone calls, electronic health records [Hauskrecht et al., 2010], etc. Long sequences are usually mined using a *sliding window* approach, where a window of a specific width is slid along the sequence and only patterns that are observed within this window are considered to be valid [Mannila et al., 1997, Höppner, 2003, Moerchen, 2006b].

4.2 TEMPORAL DATA CLASSIFICATION

In this section, we review commonly used methods for classifying temporal data. First, let us make a distinction between temporal classification and time series forecasting. The task of temporal classification is defined as follows: "Given an unlabeled sequence or time series T, assign it to one of predefined classes". On the other hand, the task of time series forecasting is defined as follows: "Given a time series T that contains n data points, predict its future values at time $n+1, n+2, ...$". We start by briefly reviewing the most common time series forecasting methods. After that, we discuss in more details temporal classification methods, which are more related to the topic of the thesis.

In **time series forecasting**, the goal is to learn a model that can predict future values of a time series based on its past values. This area has been extensively studied in statistics [Shumway and Stoffer, 2006]. One of the most popular techniques is *Auto-Regressive Integrated Moving Average* (*ARIMA*), which fits a parametric model that approximately generates the values of the time series and uses it to predict the future values. *Generalized Auto-Regressive Conditional Heteroscedastic* (*GARCH*) is another popular method that is used to model changes of variance along time (heteroskedasticity). *Recurrent Neural Networks* (*RNN*) [Rojas, 1996] is a machine learning approach that extends neural networks to the temporal domain. The idea is to add a context node to the hidden layer to summarize the network output at time t. This node is fed as input to the network at time $t+1$, which creates a directed cycle in the network. This modification allows RNN to capture dynamic temporal behavior.

In **temporal classification**, each sequence (time series) is assumed to belong to one of finitely many predefined classes and the goal is to learn a model that can classify future sequences. There are many practical applications of temporal classification, such as classifying Electroencephalography signals [Xu et al., 2004], speech recognition [Rabiner, 1989], gesture recognition [Li et al., 2009], handwritten word recognition [Plamondon and Srihari, 2000] and more. In the following, we describe the main temporal classification approaches.

4.2.1 The Transformation-based Approach

This approach applies a space transformation on the original time series and learn the classification model in the transformed space. It is mostly used when the data instances are *univariate numeric time series*.

The most straightforward transformation is to represent every time series of length n as a point in an n-dimensional Euclidean space and then apply standard machine learning methods. However, this approach is often limited because it completely ignores correlations between the consecutive time points, which is usually very informative for classification.

To overcome this shortcoming, a transformation can be applied to de-correlate the features (the time series points) and reveal structures that are often hidden in the time domain. In [Batal and Hauskrecht, 2009], time series are first transformed from the time domain into a frequency domain using *Discrete Cosine Transform* (*DCT*), or into a joint time-frequency domain using *Discrete Wavelet Transform* (*DWT*). After that, feature selection is performed on the transformed representation in order to select the classification features. The experiments showed that this method is able to outperform classifiers that are built using the original time series (without any transformation). [Ye and Keogh, 2009] introduced a transform based on *shapelets*, which are are time series subsequences that are in some sense maximally representative of of class membership.

4.2.2 The Instance-based Approach

This approach stores all training instances and defers the classification of a new instance until the time it is encountered (lazy classification). The most common instance-based method is *k-nearest neighbor* (*KNN*), which classifies an instance by taking a majority vote of its k nearest neighbors.

The success of KNN heavily depends on the quality of the distance metric. For time series data, using simple metrics such as the Euclidean distance may result in poor classification performance, especially when time series from the same class are similar in shape, but are not perfectly aligned on the time axis (with phase shifts). To overcome this, we can use the *Dynamic Time Warping* (*DTW*) [Ratanamahatana and Keogh, 2005], which has

the ability to warp the time axis of the compared time series in order to achieve a better alignment. The work by [Xi et al., 2006] shows that the 1NN classifier with DTW is an exceptionally accurate method for classifying *univariate numeric time series*, such as the benchmark data provided by [Keogh et al., 2010].

For classifying *univariate symbolic sequences*, such as gene sequences or protein sequences, we can apply KNN with the *edit distance* (Levenshtein distance) [Levenshtein, 1966]. This distance is defined as the minimum number of edits needed to transform one symbolic sequence into the other, with the allowable edit operations being insertion, deletion, or substitution of a single symbol.

Note that instance-based approaches are not used for classifying *multivariate time series* or *multivariate sequences*. The reason is that it is very difficult to define a meaningful distance metric to compare instances for such data.

4.2.3 The Model-based Approach

This approach learns a parametric model (usually a probabilistic model) from the training data and uses it for classification. The most common model-based method for temporal classification is the *Hidden Markov Model* (*HMM*) [Rabiner, 1989]. HMM is very popular for many applications, such as speech recognition, gesture recognition, part-of-speech tagging and bioinformatics. The common way to use HMM for temporal classification is to apply the following two phases [Blasiak and Rangwala, 2011]:

- *Training phase:* Learn a separate model for each class label using the Baum-Welch algorithm.
- *Testing phase:* Classify the test sequence by evaluating its probability under the different models (using the forward algorithm) and assigning it to the class corresponding to the most probable model.

The *Conditional Random Field* (*CRF*) model [Lafferty et al., 2001] is a discriminative model that conditions on the entire observation sequence. It is more flexible that HMM because it avoids the need for independence assumptions between observations. CRF has been recently applied for temporal classification. [Vail et al., 2007] showed that CRF outperforms

HMM on an activity recognition task.

HMM and CRF can be applied on *univariate symbolic sequences*, *univariate time series* and *multivariate time series*. However, both models assume that the observations are equally spaced in time. Hence, they cannot be applied on *irregularly sampled* temporal data, such as the electronic health records data.

4.2.4 The Pattern-based Approach

In Section 2.4.4, we discussed several pattern mining methods for classifying atemporal data. Here, we discuss pattern mining methods for classifying temporal data. We describe methods that classify *symbolic sequences* (time point data). We are not aware of any pattern-based method that was proposed to classify *state sequences* (time interval data), which is the focus of our work.

The methods by [Tseng and Lee, 2005, Exarchos et al., 2008] classify symbolic sequences by applying a *two-phase* approach, which mines all frequent sequences (sequential patterns) in the first phase and selects the classification sequences in the second phase. As opposed to the two-phase approach, the recently proposed method by [Ifrim and Wiuf, 2011] smartly interleaves pattern selection and frequent pattern mining. It employs gradient-bounded coordinate descent to efficiently select discriminative sequences without having to explore the whole space of subsequences. Their experiments showed that this method can achieve comparable performance to the state of the art kernel-based support vector machine methods (e.g, [Leslie et al., 2002]) for classifying symbolic sequences.

This thesis focuses on the pattern-based approach. The next two sections discuss in more details the algorithms for mining time point data and time interval data.

4.3 TEMPORAL PATTERNS FOR TIME POINT DATA

In this section, we review the major approaches for mining *symbolic sequences*, where the events are *instantaneous* and do not have time durations.

4.3.1 Substring Patterns

The simplest form of patterns that can be extracted from time point symbolic sequences are the **substring patterns** [Manber and Myers, 1990, Fischer et al., 2008], which are subsequences of symbols that appear consecutively in a sequence (without gaps). For example, Figure 21 shows the occurrences of substring pattern $< B, C, D >$ in a symbolic sequence. Discovering such patterns is mostly used in bioinformatics and computational biology for matching sequences of nucleotides or amino acids. Substring pattern mining is applied on *univariate symbolic sequences* that are *regularly sampled in time*.

Figure 21: An example showing the occurrences of substring pattern $< B, C, D >$ in a symbolic sequence.

4.3.2 Sequential Patterns

Sequential patterns are more general than substring patterns because they do not have to be consecutive in the sequence (allow gaps). Similar to itemset mining, sequential pattern mining was initially proposed for analyzing market basket data and customer shopping behavior [Agrawal and Srikant, 1995]. An example of a sequential pattern is "customers who buy a Canon digital camera are likely to buy an HP color printer within a month".

In the following example, we illustrate the main concepts of sequential pattern mining in the setting of market basket data. However, the presented concepts generalize to other domains, such as telecommunication data, machine logs, web-click data and more.

Example 13. *Consider the data in Table 9, where the alphabet of items is* $\Sigma = \{A, B, C, D\}$ *and there are 5 transactions* T_1 *to* T_5*. Each transaction is a sequence of events (customer visits to the supermarket) and each event can be a single item or a set of items (items bought*

by the customer on the same trip to the supermarket). For example, customer T_1 bought first item A, then item C, then items B and A together. We can see that sequential pattern $P_1 =< C,B >$ (C followed by B) appears in transactions T_1, T_2, T_3 and T_4, hence $sup(P_1) = 4$, while sequential pattern $P_2 =< B,C >$ appears only in transaction T_4 and hence $sup(P_2) = 1$.

Transaction	sequence of items
T_1	$< A,C,(B,A) >$
T_2	$< C,(D,A),B >$
T_3	$< A,C,B,A >$
T_4	$< C,(B,D),(A,C) >$
T_5	$< B,D >$

Table 9: An example of a sequence data.

The standard sequential pattern mining framework only cares about the order of events rather than their exact times. Therefore, sequential pattern mining does not require the original sequences to be *regularly sampled in time*. Note that the application of sequential pattern mining goes far beyond the market basket analysis task. It can be applied to any kind of *univariate* or *multivariate symbolic sequences*.

In the following, we first outline the most common sequential pattern mining algorithms and then discuss how to reduce the number of sequential patterns using temporal constraints.

Mining algorithms: The first algorithm for mining frequent sequential patterns was proposed by [Agrawal and Srikant, 1995], which is based on the Apriori approach (see Section 2.2.1). *PrefixSpan* [Pei et al., 2001] mines sequential patterns using the pattern growth approach (see Section 2.2.2) and *SPADE* [Zaki, 2001] uses the vertical data approach (see Section 2.2.3). *CloSpan* [Yan et al., 2003] and *BIDE* [Wang and Han, 2004] are two efficient methods for mining closed sequential patterns (see Section 2.3.1 for the definition of closed patterns).

Temporal Constraints: Mining the complete set or even the closed set of frequent sequential patterns usually leads to results that are too large to be analyzed by humans.

Many of the concise representations described in Section 2.3 can be applied for compressing sequential patterns. Another way to reduce the number of sequential patterns is to impose **temporal constraints** on the patterns. One common temporal constraint is to restrict the *total duration* of the pattern. For example, we may specify that the total pattern duration must not exceed w time unites (e.g., 6 months). This constraint translates to defining a sliding window of width w and mining only sequential patterns that can be observed within this window. Another common temporal constraint is to define the *maximum gap* that is allowed between consecutive events in a pattern. For example, we may specify that the difference between consecutive events should not be more than g time units (e.g., 2 weeks). Incorporating temporal constraints in the Apriori approach is described in [Srikant and Agrawal, 1996] and in the pattern growth approach is described in [Pei et al., 2007].

4.3.3 Episode Patterns

We saw that sequential patterns are used to express order among events. **Episode patterns** [Mannila et al., 1997, Méger and Rigotti, 2004] are more general than sequential patterns because they can also express the concept of *concurrency*. [Mannila et al., 1997] defined two special types of episodes:

1. **Serial episodes**: express order of events (equivalent to a sequential pattern with a maximum pattern duration constraint).
2. **Parallel episodes**: express concurrency of events (the order does not matter).

In general, an episode is a *combination of serial and parallel episodes*. For example, an episode can be a sequence of parallel episodes or a parallel combination of serial episodes. Episodes are usually represented as a directed acyclic graph of events whose edges specify the temporal order of events. On the right of Figure 22, we show an example of an episode pattern that is a parallel combination of two serial episodes $< F_2 = C,\ F_1 = A >$ and $< F_3 = D,\ F_3 = B >$. Note that this episode represents only a partial order because the relation between $F_2 = C$ and $F_3 = D$ is unspecified. On the left of Figure 22, we show the occurrences of this episode in a multivariate symbolic sequence using a sliding window.

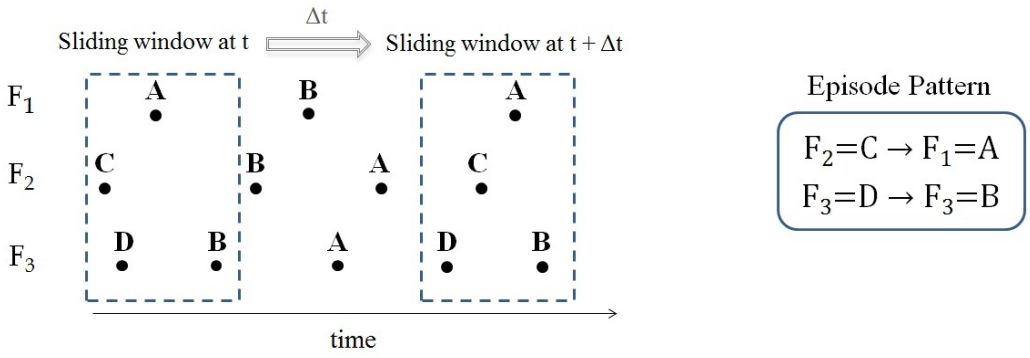


Figure 22: An example showing the occurrences of episode pattern ($< F_2 = C, F_1 = A >, < F_3 = D, F_3 = B >$) in a multivariate symbolic sequence using a sliding window.

Although these episodes are able to represent partial order patterns, not every partial order pattern can be represented as a combination of serial and parallel episodes. [Casas-Garriga, 2005] proposed an algorithm for mining *all partial order patterns*. However, their algorithm is computationally very expensive and does not scale up to large data.

4.4 TEMPORAL PATTERNS FOR TIME INTERVAL DATA

In this section, we review the main approaches for mining *state sequences*, where the events have *time durations*.

4.4.1 Allen's Temporal Relations

The temporal relation between two time points can be easily described using the following three relations: *before*, *equals* (at the same time) and *after*. However, when the events have time durations, the relations become more complex. The most common way is to use **Allen's temporal relations** [Allen, 1984], which were introduced in artificial intelligence for temporal reasoning and have been later adopted in the research for mining time interval

data [Kam and Fu, 2000, Höppner, 2003, Papapetrou et al., 2005, Winarko and Roddick, 2007, Patel et al., 2008, Moskovitch and Shahar, 2009].

Allen formalized a logic on time intervals (states) by specifying 13 different temporal relations and showing their completeness. Any pair of states are temporally related by one of the following 13 relations: *before*, *meets*, *overlaps*, *is-finishes-by*, *contains*, *starts*; their corresponding inverses: *after*, *is-met-by*, *is-overlapped-by*, *finishes*, *during*, *is-started-by* and the *equals* relation (see Figure 23).

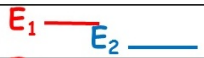
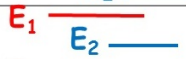
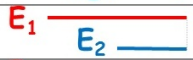
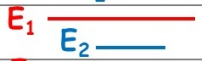
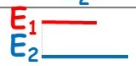
	E_1 before E_2	E_2 after E_1
	E_1 meets E_2	E_2 is-met-by E_1
	E_1 overlaps E_2	E_2 is-overlapped-by E_1
	E_1 is-finished-by E_2	E_2 finishes E_1
	E_1 contains E_2	E_2 during E_1
	E_1 starts E_2	E_2 is-started-by E_1
	E_1 equals E_2	E_2 equals E_1

Figure 23: Allen's temporal relations.

Allen's relations can precisely describe the temporal relation between two states. However, it is less obvious how to describe the temporal relations for a pattern with multiple states. In the following, we outline the most common time interval pattern representations and discuss several mining algorithms. We will see that most of these algorithms are extensions of sequential pattern mining algorithms [Srikant and Agrawal, 1996, Zaki, 2001, Pei et al., 2001, Yan et al., 2003, Lin and Lee, 2005] to handle time intervals[3].

In order to simplify the notations in the subsequent sections, we use a subscript next to each state to denote its temporal variable. For example, in Figure 24, state A_1 means $F_1 = A$, state D_2 means $F_2 = D$ and state B_3 means $F_3 = B$.

[3]Sequential pattern mining is a special case of time interval pattern mining, in which all intervals are instantaneous with zero durations.

4.4.2 Early Approaches

[Villafane et al., 2000] is the earliest work in the area of mining time interval patterns. Their temporal patterns are restricted to having *only containment relations*, which corresponds to Allen's *contains* relation. An example of such patterns is "during a FLU infection, a certain strain of bacteria is often found on the patient".

[Kam and Fu, 2000] is the first to propose using Allen's relations for defining temporal patterns. Their temporal patterns, called the **A1 patterns**, are based on a *nested representation* which only allows concatenations of temporal relations on the right hand side of the pattern. For example, $P_1 = ((A_1\ before\ D_2)\ \ overlaps\ B_3)$ is interpreted as: "state A_1 is before state D_2 and the interval that contains both A_1 and D_2 overlaps with state B_3". Figure 24 shows a graphical representation of a situation that matches pattern P_1. Note that this representation is *ambiguous* because the situation in Figure 24 can be also described as: $P_2 = ((A_1\ overlaps\ B_3)\ contains\ D_2)$ ("state A_1 overlaps state B_3 and the interval that contains both A_1 and B_3 contains state D_2"). That is, the exact same situation in the data can be described by multiple A1 patterns, which is an undesirable property.

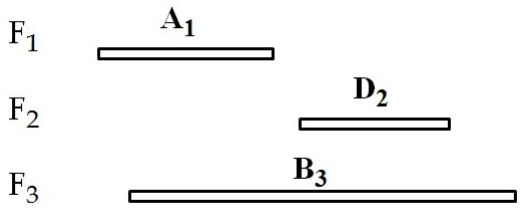


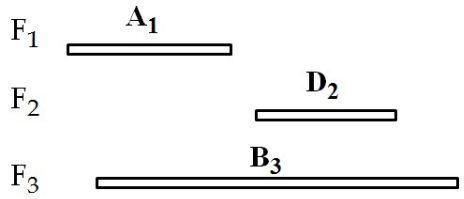
Figure 24: The pattern in this figure can be described by several A1 patterns, such as $((A_1\ before\ D_2)\ \ overlaps\ \ B_3)$ and $((A_1\ overlaps\ B_3)\ \ contains\ D_2)$. This illustrates the ambiguity of the nested representation.

4.4.3 Höppner Representation

[Höppner, 2003] proposed the first *non-ambiguous* representation for defining time interval patterns. The idea is to first define the **normalized form** of temporal patterns, where the

states of a pattern are always sorted in increasing index according to their start times, end times and value[4]. Now in order to define a temporal pattern with k states (a *k-pattern*), we should specify the relations for all pairs of states. However, since the patterns are in the normalized form, we only need 7 of the 13 Allen's relations, namely: *before*, *meets*, *overlaps*, *is-finished-by*, *contains*, *starts* and *equals*.

Figure 25 shows an example of a temporal pattern with states A_1, B_3 and D_2. The matrix on the right of the figure specifies the relations for all pairs of states. Because the pattern is in normalized form, it suffices to only specify the relations between each state and all of its following states (only using the upper triangular matrix). Therefore, defining a *k-pattern* requires specifying all of its k states and specifying all $k \times (k-1)/2$ relations between the states. Hence the "cost" of this representation is $k + k \times (k-1)/2$, where the cost is informally defined as the effort needed to describe a pattern to the user.



	A_1	B_3	D_2
A_1	Equals	Overlaps	Before
B_3	Overlapped-by	Equals	Contains
D_2	After	During	Equals

Figure 25: A temporal pattern represented using Höppner's non-ambiguous representation.

For mining these patterns, [Höppner, 2003] used a sliding window approach to extract local temporal patterns (i.e., patterns with limited total durations) from a single long multivariate state sequence (weather data). He defined the support of a pattern to be the *total time in which the pattern can be observed within the sliding window*. Note that this definition is different from the conventional support definition, which is the number of times a pattern appears in the data. His algorithm extends Apriori for sequential patterns [Agrawal and Srikant, 1995] to handle the more complex case of time interval patterns.

Later on, the same pattern representation as [Höppner, 2003] was independently de-

[4]We first sort all states by their start times. If two states have the same start time, we sort them by their end times. If they also have the same end time, we sort them alphabetically.

scribed by [Papapetrou et al., 2005, Winarko and Roddick, 2007]. [Papapetrou et al., 2005] proposed the hybrid-DFS method, which uses a tree-based enumeration algorithm like the one introduced by [Bayardo, 1998]. [Winarko and Roddick, 2007] proposed the ARMADA algorithm, which extends a sequential pattern mining algorithm called MEMISP [Lin and Lee, 2005] to mine time interval patterns.

4.4.4 Other Representations

[Moerchen, 2006a, Moerchen, 2006b] proposed the *Time Series Knowledge Representation* (***TSKR***) as an alternative to using Allen's relations. This representation is based on the concept of *cords*, which describe coincidence (overlapping) of several states. Figure 26 shows the same pattern in Figure 25 in the TSKR format. The cords of this pattern are $< A_1, A_1B_3, B_3, D_2B_3, B_3 >$, which mean that state A_1 is first observed alone, then both A_1 and B_3 are observed, then B_3 alone and so on. Mining TSKR patterns was done by modifying the CloSpan algorithm [Yan et al., 2003] and using a sliding window approach. Moerchen also proposed mining patterns that describe partial order of cords, which he called *phrases*.

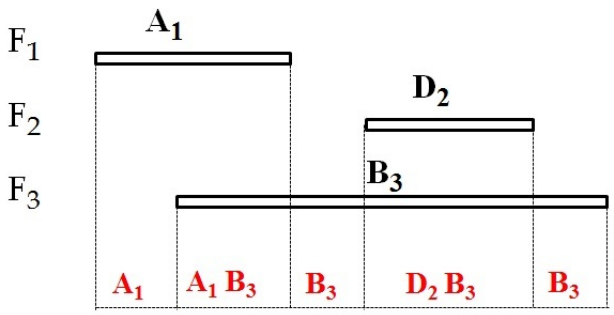


Figure 26: A temporal pattern represented using the TSKR representation.

In [Sacchi et al., 2007], the user is required to define beforehand a set of complex patterns of interest, which are called *Abstractions of Interest* (***AoI***). These *AoI* constitute the basis for the construction of temporal rules. A temporal association rule is defined as $A \Rightarrow c$, where A is a set of *AoI* and c is a single *AoI*. The temporal rule holds when the *precedes* re-

lation is satisfied between the antecedent (A) and the consequent (c). This relation requires specifying three parameters (see Figure 27):

1. *Left shift*: the maximum allowed time between the start of c and the maximum start of the patterns in A.
2. *Gap*: the maximum allowed time between the start of c and the minimum end of the patterns in A.
3. *Right shift*: the maximum allowed time between the end of c and the minimum end of the patterns in A.

As we can see, this representation relies on the user to define the *AoI* and to specify the three parameters of the *proceeds* relation.

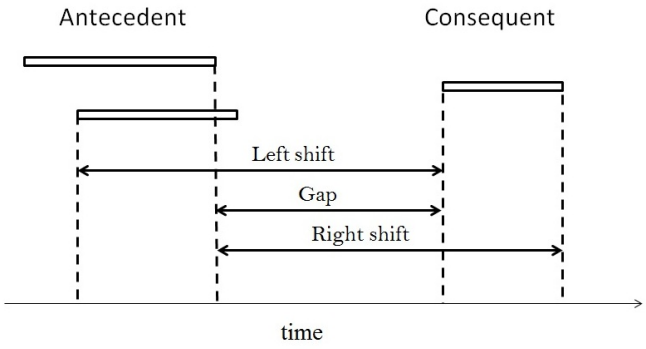


Figure 27: Illustrating the three parameters of the *precedes* relation for a rule that has two *AoI* in the antecedent.

[Patel et al., 2008] realized the ambiguity of the nested representation of [Kam and Fu, 2000] and proposed augmenting this representation with additional counts to make it unambiguous. However, the resulting patterns are very complicated and hard to interpret. Besides, there was no clear justification (nor a comparison) for using this representation instead Höppner's representation that is already non-ambiguous and easier to understand.

[Wu and Chen, 2007] proposed a very different interval pattern language, where every state S in the data is represented by its two interval boundaries: 1) the time S appears, denoted as S^+ and 2) the time S disappears, denoted by S^-. An interval pattern is repre-

sented as a sequence of its state boundaries. Because this representation only deals with time points (as opposed to time intervals) and sorts the boundaries according to their times, it suffices to use only two relations: *before* (<) and *equals* (=) to describe any temporal pattern.

Figure 28 shows the same pattern in Figure 25 represented as $A_1^+ < B_3^+ < A_1^- < D_2^+ < D_2^- < B_3^-$. As we can see, representing a *k-pattern* in this framework requires specifying the k states and $2 \times k - 1$ relations between their boundaries. Hence, the cost of this representation is $k + 2 \times k - 1$. This representation can be considered more compact than Höppner's representation (which has a cost of $k + k \times (k-1)/2$) when the patterns are large. However, it may be less intuitive and harder to communicate with the experts.

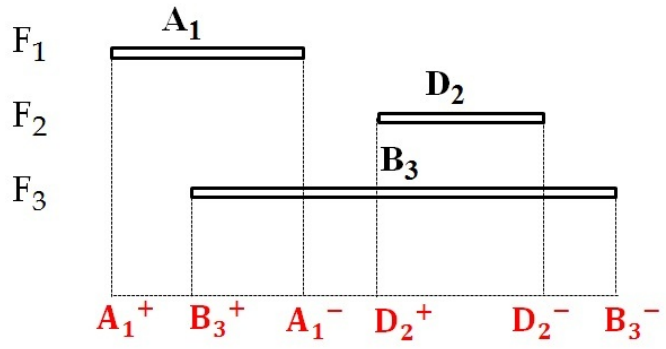


Figure 28: A temporal pattern represented by the order of its state boundaries.

In order to mine these patterns, the algorithm by [Wu and Chen, 2007] first converts all state sequences in the data into sequences of state boundaries and then employs a sequential pattern mining algorithm. This algorithm is modification of PrefixSpan [Pei et al., 2001] that adds additional constraints to ensure that the sequential state boundary patterns correspond to valid time interval patterns. For example, patterns like $A_1^- < A_1^+$ or $A_1^+ < B_2^- < A_1^-$ will be suppressed from the results.

It is important to note that there is a one-to-one correspondence between this representation and Höppner's representation. In other words, both representations give the same result if they are applied on the same data with the same minimum support.

The work of [Moerchen and Fradkin, 2010] is similar to [Wu and Chen, 2007] in that they represent the states by their boundaries. However, the difference is that [Moerchen and Fradkin, 2010] do not force the mined patterns to be proper time interval patterns, which results in what they call *Semi-Interval Sequential Patterns* (***SISP***). According to their definition, patterns like $A_1^- < A_1^+$ or $A_1^+ < B_2^- < A_1^-$ are considered to be valid results.

As an example, consider $A_1^+ < B_3^+ < A_1^- < D_2^+$. This SISP matches all three Höppner's patterns in Figure 29. As we can see, SISPs are always a super set of Höppner's patterns[5], which the authors claim is the advantage of this representation (a more flexible representation).

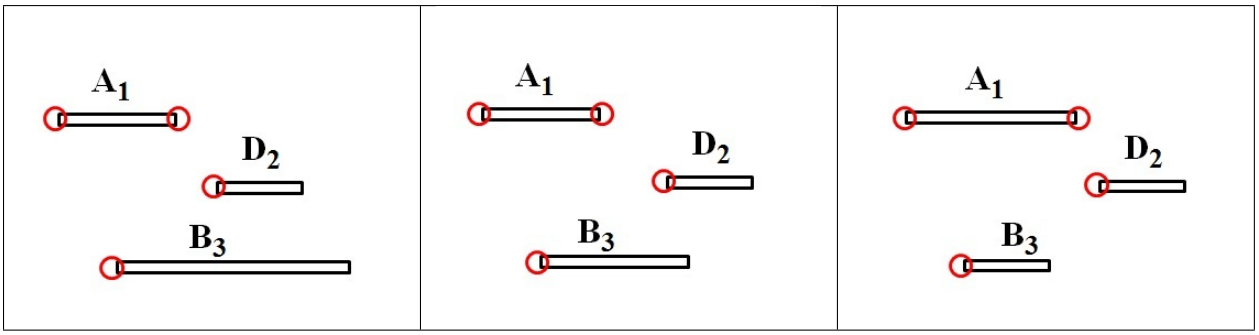


Figure 29: Three different Höppner's patterns that match *SISP*: $A_1^+ < B_3^+ < A_1^- < D_2^+$.

However, using this representation has the following disadvantages:

1. It usually returns a huge number of patterns[6].

2. Patterns do not describe precise situations in the data (Figure 29). For example, SISP $A_1^- < B_2^-$ does not tell us much about the temporal relation between state A_1 and state B_2. It is possible that B_2 *contains* A_1, A_1 *overlaps* B_2 or A_1 *before* B_2, which are conceptually very different situations.

3. The mining is computationally very expensive because the search space is very large.

[5] A Höppner's pattern matches only one SISP, while a SISP can match several Höppner's patterns.

[6] As we saw in Section 2.3, one of the most active research in pattern mining is reducing the number of discovered patterns. This representation blows up the number of temporal patterns, which makes the task of knowledge discovery much harder.

4.5 TEMPORAL ABSTRACTION

Temporal abstraction [Shahar, 1997] transforms numeric time series from a low-level quantitative form into a high-level qualitative form. More specifically, temporal abstraction segments the numeric time series into a sequence of states, where each state represents a property that holds during a time interval. These states become the building blocks to construct more complex time interval patterns as we explained in the previous section. That is, temporal abstraction can be seen as a pre-processing step for time interval pattern mining. Usually, we would like these abstractions to be meaningful and easy to interpret by humans so that the resulting patterns would be useful for knowledge discovery. In the following, we describe the three main temporal abstraction approaches: abstraction by clustering, trend abstractions and value abstractions.

4.5.1 Abstraction by Clustering

Abstraction by clustering is the process of *inductively* deriving a set of states for a numeric time series by clustering similar parts of the time series [Das et al., 1998, Kadous, 1999]. In this approach, subsequences (portions of time series) are considered data objects and clusters correspond to inductively derived states that describe similar subsequences. More concretely, given a time series $T = (T_1, .., T_n)$ and a window of width w, abstraction by clustering works as follows:

1. Extract all subsequences of length w using a sliding window.
2. Cluster the set of all subsequences to obtain clusters $C_1, ..., C_k$. Note that this requires defining a distance metric for measuring similarity between two subsequences of length w (e.g., the Euclidean distance or dynamic time warping [Ratanamahatana and Keogh, 2005]).
3. Define the abstracted version of T by assigning every subsequence to its corresponding cluster symbol: $T' = (s_{j(1)}, s_{j(2)}, ..., s_{j(n-w+1)})$, where s_i is the symbol for cluster C_i.

The basic property of abstraction by clustering is that the alphabet is not predefined, but automatically derived from the data. However, it has the following disadvantages:

1. The output highly depends on the choice of the *window width* w, the choice of the *distance metric* and the choice of the *clustering algorithm*.
2. The resulting abstractions are usually very hard to understand by humans because they do not have intuitive interpretations.

4.5.2 Trend Abstractions

Trend abstractions is the process of partitioning numeric time series based on its local trends. This is usually done using the following steps:

1. Apply *Piecewise Linear Representation* (***PLR***) to approximate the series with straight lines (Figure 30).
2. Define the abstractions based on the slopes of the fitted lines (segments). For example, assume we want to use alphabet Σ = { *decreasing*, *steady*, *increasing* }. We label the segments that result from applying PLR as following: Given a threshold δ, if the slope of the segment is positive and greater than δ, we assign it the label *increasing*. If the slope is negative and its absolute value is greater than δ, we assign it the label *decreasing*. Otherwise, we assign it the label *steady*.

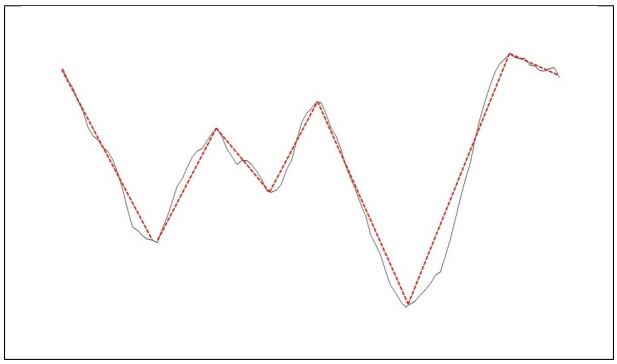


Figure 30: An example illustrating the piecewise linear representation.

PLR has been also used to support a variety of data mining tasks, such as classification [Batal et al., 2009, Batal et al., 2011], clustering [Keogh and Pazzani, 1998] and indexing [Keogh et al., 2001].

The algorithm for finding a *PLR* is usually called a *segmentation algorithm*. It requires defining a measure to evaluate the fitness of a potential segment. The most common measure is the residual error, which is the sum of squares of the differences between the actual time series values and the fitted segment.

The objective of a segmentation algorithm can be formulated in one of the following two ways:

1. Produce a *PLR* such that the error for any segment does not exceed a user-specified threshold ϵ.
2. Produce the "best" *PLR* using only k segments.

[Keogh et al., 1993] group time series segmentation algorithms into the following three categories:

1. *Sliding window:* This approach works by first anchoring the left point of a potential segment at the first time point of the time series, then attempting to approximate the data to its right with increasing longer segments. At some point i, the error for the potential segment becomes greater than ϵ, so the subsequence from the anchor to point $i-1$ is transformed into a segment. The anchor is then moved to location i and the process repeats until the entire time series is segmented. This algorithm is attractive because it is very simple, intuitive and online in nature.
2. *Top-down:* This approach works by considering every possible two-ways partitioning of the time series and splitting it at the best location. Both segments are tested to check whether the stopping condition is satisfied (the errors of all segments is below ϵ or the total number of segments is k). If not, the algorithm recursively continues splitting the subsequences until the stopping condition is satisfied.
3. *Bottom-up:* This approach begins by creating the finest possible segmentation of the time series, so that $n/2$ segments are used to approximate a time series of length n. Next, the cost of merging each pair of adjacent segments is calculated and the algorithm iteratively merges the lowest cost pair until the stopping condition is satisfied.

Note that both top-down and bottom-up allow the user to specify the maximum error per segment ϵ or to specify the desired number of the segments k. On the other hand, sliding

window only allows the first option.

4.5.3 Value Abstractions

Value abstractions is the process of partitioning numeric time series based on its values. The most straightforward approach is to use standard discretization methods [Yang et al., 2005], such as equal width histogram or equal frequency histogram, to discretize the time series values. However, such methods ignore the temporal order of values, which may lead to states that are not practically meaningful.

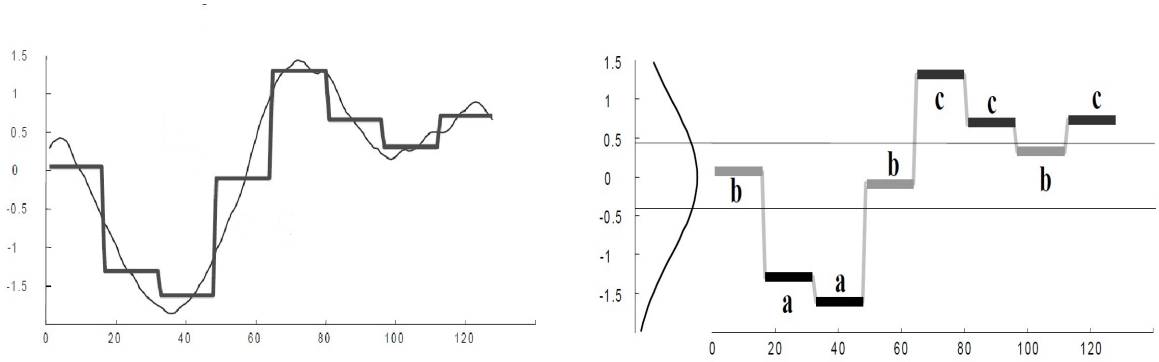
To overcome this, [Moerchen and Ultsch, 2005] argued that a good discretization method should produce states with persisting behavior. They proposed the *Persist* algorithm, which tries to obtain discretization symbols with a self transition distribution (under the first order Markovian assumption) $P(S_i = s|S_{i-1} = s')$ significantly higher than their marginal distribution $P(S_i = s)$.

Symbolic Aggregate Approximation (*SAX*) [Lin et al., 2003] is a popular time series discretization technique, which applies the following two steps:

1. Divide the time series into q *equal-sized* segments and replace the values in each segment with their average. These q coefficients are known in the literature as the *Piecewise Aggregate Approximation* (*PAA*) [Keogh et al., 2000].

2. Convert the *PAA* coefficients to k symbols (k is the alphabet size) by determining the breakpoints which divide the distribution space into k *equiprobable* regions. In other words, determine the breakpoints such that the probability of a segment falling into any of the regions is approximately the same. Once the breakpoints are determined, each region is assigned a symbol.

Figure 31:a shows the *PAA* of a time series using $q = 8$ segments. Figure 31:b shows the *SAX* symbols for an alphabet of size $k = 3$.



(a) (b)

Figure 31: An example illustrating the SAX representation. The figure on the left shows the *PAA* of the series and the figure on the right shows the derived *SAX* symbols.

4.6 SUMMARY

Temporal data mining refers to mining data that have a temporal aspect. Temporal data can take a variety of forms from simple univariate regularly sampled data, to complex multivariate irregularly sampled data. In this chapter, we presented an overview of temporal data mining and we mostly focused on temporal classification and temporal pattern mining.

The goal of temporal classification is to train a classifier that can accurately predict the class label for future unlabeled sequences. There are several approaches to construct a temporal classifier:

1. Apply a time series transformation such as *DFT*, *DWT* or *shapelet* transform. This approach is mostly applied to univariate regularly sampled numeric time series.
2. Define a distance metric and classify unlabeled sequences according to the most similar training sequences (the KNN classifier). This approach is mostly applied to univariate regularly sampled series.
3. Learn a probabilistic model such as *HMM* or *CRF* and use it for classification. This approach can handle univariate and multivariate regularly sampled series.
4. Apply temporal pattern mining and use the patterns for classification.

Temporal pattern mining can be applied to symbolic sequences to mine sequential patterns (Section 4.3) and to state sequences to mine time interval patterns (Section 4.4). In order to apply temporal pattern mining on numeric time series, the series should be converted into state sequences using temporal abstraction (Section 4.5).

5.0 MINING PREDICTIVE TEMPORAL PATTERNS

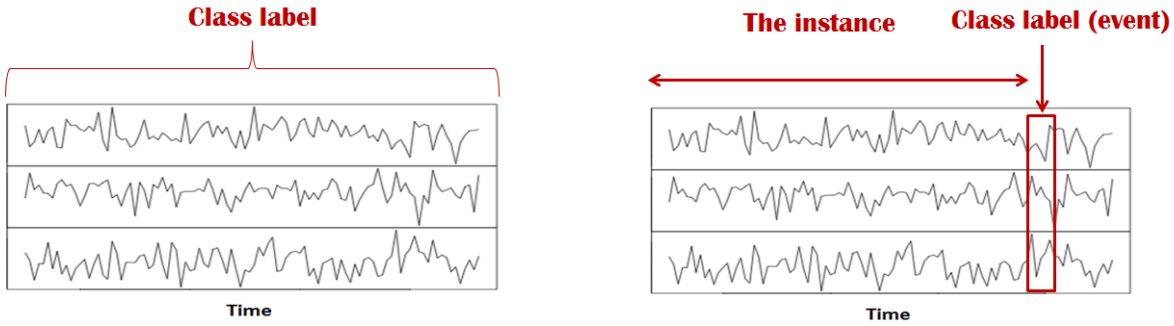
The majority of existing temporal classification methods (see Section 4.2) assume that each data instance (represented by a single or multiple time series) is associated with a single class label that affects its *entire* behavior. That is, they assume that all temporal observations are potentially equally useful for classification.

However, the above assumption is not suitable when considering *event detection* problems. In these problems, a class label denotes an event that is associated with a specific time point (or a time interval) in the instance, not necessarily the entire instance (see Figure 32). The objective is to learn a predictive model that can accurately identify the occurrence of events in unlabeled time series instances (a monitoring task). Examples of such problems are the detection of adverse medical events (e.g. drug toxicity) in clinical data [Hauskrecht et al., 2010], the detection of equipment malfunctions [Guttormsson et al., 1999], fraud detection [Srivastava et al., 2008], environmental monitoring [Papadimitriou et al., 2005], intrusion detection [Chandola et al., 2006] and others.

Given that class labels are associated with specific time points (or time intervals), each time series instance can be annotated with multiple labels[1]. Consequently, the context in which the classification is made is often local and affected by the most *recent behavior* of the monitored instances.

This chapter proposes a novel temporal pattern mining approach for event detection that takes into account the local nature of decisions. We present the ***Recent Temporal Pattern* (*RTP*)** mining framework, which mines frequent temporal patterns backward in time, starting from patterns related to the most recent observations. Applying this tech-

[1] In the clinical domain, a patient may be healthy at first, then develop an adverse medical condition, then be cured and so on.



(a) (b)

Figure 32: Illustrating the difference between temporal classification and event detection.

nique, temporal patterns that extend far into the past are likely to have low support in the data and hence would not be considered for classification. Incorporating the concept of recency in temporal pattern mining is a new research direction that, to the best of our knowledge, has not been previously explored in the pattern mining literature.

We primarily focus on applying RTP mining to temporal data in Electronic Health Record (EHR) systems. In this data, each record (data instance) consists of multiple time series of clinical variables collected for a specific patient, such as laboratory test results and medication orders. The record may also provide information about patient's diseases and other adverse medical events over time. Our objective is to mine patterns that can accurately predict adverse medical events and apply them to monitor future patients. This task is extremely used for intelligent patient monitoring, outcome prediction and decision support.

The task of temporal modeling for EHR data is challenging because the data are *multivariate* and the time series for clinical variables are *irregularly sampled in time* (measured asynchronously at different time moments). Therefore, most existing times series classification methods [Blasiak and Rangwala, 2011, Vail et al., 2007], time series similarity measures [Ratanamahatana and Keogh, 2005, Yang and Shahabi, 2004] and time series feature extraction methods [Li et al., 2010, Batal and Hauskrecht, 2009, Weng and Shen, 2008]

cannot be directly applied on the raw EHR data.

The key step for analyzing EHR data is defining a language that can adequately represent the temporal dimension of the data. Our approach relies on 1) temporal abstractions (see Section 4.5) to convert numeric time series variables to *time interval* sequences and 2) temporal relations (see Section 4.4.1) to represent temporal interactions among the variables. For example, this allows us to define complex *temporal patterns* (time-interval patterns) such as "the administration of heparin precedes a decreasing trend in platelet counts".

After defining temporal patterns, we need to design an efficient algorithm for mining predictive patterns in temporally abstracted time interval data. This task is very challenging because the search space of such patterns is extremely large. All existing methods in this area (see Section 4.4) have been applied in an unsupervised setting to mine temporal association rules [Moskovitch and Shahar, 2009, Wu and Chen, 2007, Winarko and Roddick, 2007, Papapetrou et al., 2005, Moerchen, 2006b, Höppner, 2003]. These methods are known to be computationally very expensive and they do not scale up to large data.

In contrast to the existing methods, our work applies temporal pattern mining in the supervised setting to find patterns that are important for the event detection task. To achieve this, we present an efficient algorithm for mining RTPs (see above) from time interval data. We also present the *Minimal Predictive Recent Temporal Patterns* (*MPRTP*), which extends the MPP framework to the temporal domain in order to find predictive and non-spurious RTPs.

The rest of the chapter is organized as follows. Section 5.1 describes the event detection problem and briefly outlines our approach for solving it. Section 5.2 describes temporal abstraction and defines temporal patterns for time interval data. Section 5.3 defines the recent temporal patterns (RTPs) and illustrate their properties. Section 5.4 describes our algorithm for mining frequent RTPs. Section 5.5 describes our algorithm for mining minimal predictive recent temporal patterns (MPRTPs). Section 5.6 illustrates how to obtain a feature vector representation of the multivariate time series data in order to learn the event detection classifier. Section 5.7 compares our methods with several baselines on a synthetic dataset and two real-world clinical datasets. Finally, Section 5.8 summarizes the chapter.

5.1 PROBLEM DEFINITION

Let $D = \{< \boldsymbol{x_i}, y_i >\}_{i=1}^{n}$ be a training dataset such that $\boldsymbol{x_i} \in X$ is a multivariate temporal instance up to some time t_i and $y_i \in Y$ is a class label (an event) associated with $\boldsymbol{x_i}$ at time t_i (see Figure 32:b). The objective is to learn a function $f : X \rightarrow Y$ that can label unlabeled time series instances. This setting is applicable to a variety of event detection problems, such as the ones described in [Srivastava et al., 2008, Chandola et al., 2006, Guttormsson et al., 1999, Papadimitriou et al., 2005].

In this work, we test our method on data from Electronic Health Records (EHRs). For this task, every data instance x_i is a record for a specific patient up to some time t_i and the class label y_i denotes whether or not this patient is diagnosed with an adverse medical condition (e.g., renal failure) at t_i. Figure 33 shows a graphical illustration of an EHR instance with 3 clinical temporal variables. The objective is to learn a classifier that can predict well the studied medical condition and apply it to monitor future patients.

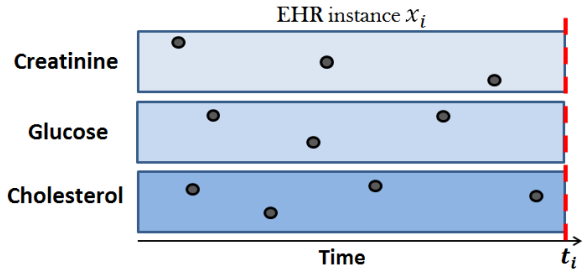


Figure 33: An example of an EHR data instance with three temporal variables. The black dots represent their values over time.

Learning the classifier directly from EHR data is very difficult because each instance consists of multiple irregularly sampled time series of different length. Therefore, we want to apply a space transformation $\psi : X \rightarrow X'$ that maps each EHR instance $\boldsymbol{x_i}$ to a fixed-size feature vector $\boldsymbol{x'_i}$, while preserving the predictive temporal characteristics of $\boldsymbol{x_i}$ as much as possible.

One approach to define ψ is to represent the data using a *predefined* set of features and

their values as in [Hauskrecht et al., 2010]. Examples of such features are "most recent creatinine value", "most recent glucose trend", "maximum cholesterol value", etc. Our approach is different because we *learn* transformation ψ from the data using temporal pattern mining. This is done by applying the following steps:

1. Convert the numeric time series variables into *time interval state sequences* using *temporal abstraction*.
2. Mine *recent temporal patterns* from the time interval data.
3. Transform each instance $\boldsymbol{x_i}$ into a *binary indictor vector* $\boldsymbol{x'_i}$ using the patterns obtained in step 2.

After applying this transformation, we can use a standard machine learning method (e.g. support vector machines, decision tree, or logistic regression) on $\{<\boldsymbol{x'_i}, y_i>\}_{i=1}^{n}$ to learn the classifier f.

In the following, we explain in details each of these steps.

5.2 TEMPORAL ABSTRACTION PATTERNS

5.2.1 Temporal Abstraction

The goal of **temporal abstraction** (Section 4.5) is to transform the numeric time series variables to a high-level qualitative form. More specifically, each temporal variable (e.g., series of platelet counts) is transformed into an *interval-based* representation $\langle v_1[s_1, e_1], ..., v_n[s_n, e_n]\rangle$, where $v_i \in \Sigma$ is an abstraction that holds from time s_i to time e_i and Σ is the abstraction alphabet that represents a finite set of permitted abstractions.

In our work, we use two types of temporal abstractions:

1. *Trend abstractions*: Segment the time series based on its local trends (see Section 4.5.2). We use the following abstractions: *Decreasing* (D), *Steady* (S) and *Increasing* (I), i.e., $\Sigma = \{D, S, I\}$. These abstractions are obtained by using the sliding window segmentation method [Keogh et al., 1993] and labeling the states according to the slopes of the fitted segments. For more information about trend segmentation, see Section 4.5.2.

2. *Value abstractions*: Segment the time series based on its values (see Section 4.5.3). We use the following abstractions: *Very Low* (*VL*), *low* (*L*), *Normal* (*N*), *High* (*H*) and *Very High* (*VH*), i.e., $\Sigma = \{VL, L, N, H, VH\}$. These abstractions are obtained using the 10th, 25th, 75th and 90th percentiles on the lab values: a value below the 10th percentile is *very low* (*VL*), a value between the 10th and 25th percentiles is *low* (*L*), and so on.

   Figure 34 shows the trend and value abstractions on a time series of platelet counts.

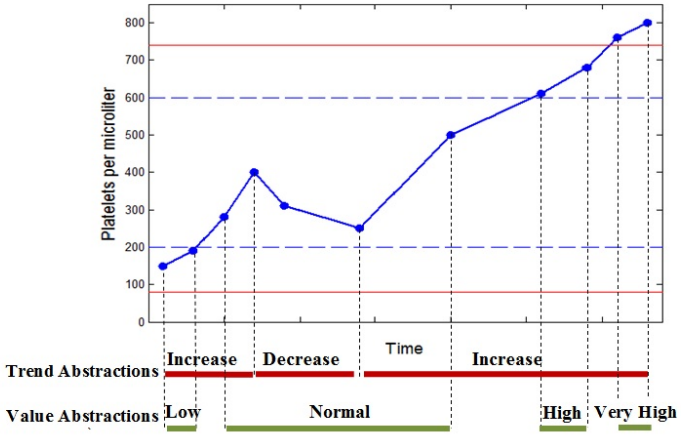


Figure 34: An example illustrating trend abstractions and value abstractions. The blue dashed lines represent the 25th and 75th percentiles of the values and the red solid lines represent the 10th and 90th percentiles of the values.

5.2.2 Multivariate State Sequences

Let a **state** be an *abstraction for a specific variable*. We denote a state S by a pair (F, V), where F is a temporal variable and $V \in \Sigma$ is an abstraction value. We sometimes denote S as $F = V$. Let a **state interval** be a *state that holds during an interval*. We denote a state interval E by a 4-tuple (F, V, s, e), where F is a temporal variable, $V \in \Sigma$ is an abstraction value, and s and e are the start time and end time (respectively) of the state interval ($E.s \leq E.e$)[2]. For example, assuming the time granularity is days, state interval $(glucose, H, 5, 10)$

[2] If $E.s = E.e$, state interval E corresponds to a time point.

represents high glucose values from day 5 to day 10.

After abstracting all time series variables, each multivariate time series instance in the data becomes a multivariate state sequence.

Definition 3. *A **Multivariate State Sequence (MSS)** is represented as a series of state intervals that are ordered according to their start times*[3]*:*

$$Z = \langle E_1, E_2, ..., E_l \rangle : \ E_i.s \leq E_{i+1}.s \quad \forall i \in \{1, ..., l-1\}$$

Note that we do not require $E_i.e$ to be less than $E_{i+1}.s$ because the state intervals are obtained from different temporal variables and their intervals may overlap. Let ***Z.end*** denote the end time of the MSS.

5.2.3 Temporal Relations

Allen's temporal logic (Section 4.4.1) describes the relations for any pair of *state intervals* using 13 possible relations. However, it suffices to use the following 7 relations: *before*, *meets*, *overlaps*, *is-finished-by*, *contains*, *starts* and *equals* because the other relations are simply their inverses. Allen's relations have been used by the majority of research on mining time interval data [Kam and Fu, 2000, Höppner, 2003, Papapetrou et al., 2005, Winarko and Roddick, 2007, Patel et al., 2008, Moskovitch and Shahar, 2009].

Most of Allen's relations require equality of one or two of the intervals end points. That is, there is only a slight difference between *overlaps*, *is-finished-by*, *contains*, *starts* and *equals* relations (see Figure 23 in Section 4.4.1). Hence, when the time information in the data is noisy (not very precise), which is the case for EHR data, using Allen's relations may cause the problem of pattern fragmentation[4] [Moerchen, 2006b].

Therefore, we opt to use only two temporal relations, namely ***before*** (***b***) and ***co-occurs*** (***c***), which we define as follows:
Given two state intervals E_i and E_j:

[3] If two state intervals have the same start time, we sort them by their end time. If they also have the same end time, we sort them by lexical order (see [Höppner, 2003]).

[4] Pattern fragmentation refers to the problem of having many different temporal patterns that describe a very similar situation in the data.

- E_i is ***before*** E_j, denoted as $\boldsymbol{b(E_i,E_j)}$, if $E_i.e < E_j.s$ (same as Allen's *before*).
- E_i ***co-occurs*** with E_j, denoted as $\boldsymbol{c(E_i,E_j)}$, if $E_i.s \leq E_j.s \leq E_i.e$. That is, E_i starts before E_j and there is a nonempty time period where both E_i and E_j occur. Note that this relation covers the following Allen's relations: *meets*, *overlaps*, *is-finished-by*, *contains*, *starts* and *equals*.

5.2.4 Temporal Patterns

In order to obtain temporal descriptions of the data, basic states are combined using temporal relations to form temporal patterns (time interval patterns). In the previous section, we defined the relation between two states to be either *before* (b) or *co-occurs* (c). In order to define relations between k states, we adopt Höppner's representation of temporal patterns (see Section 4.4.3).

Definition 4. *A* ***temporal pattern*** *is defined as* $P = (\langle S_1,...,S_k\rangle, R)$ *where* S_i *is the* i^{th} *state of the pattern and* R *is an upper triangular matrix that defines the temporal relations between each state and all of its following states:*

$i \in \{1,...,k-1\} \wedge j \in \{i+1,...,k\}$ *:* $R_{i,j} \in \{b,c\}$ *specifies the relation between* S_i *and* S_j*.*

The size of a temporal pattern P is the number of states it contains. If P contains k states, we say that P is a *k-pattern*. Hence, a single state is a *1-pattern* (a singleton). When a pattern contains only 2 states: $(\langle S_1,S_2\rangle, R_{1,2})$, we sometimes write it simply as $S_1\ R_{1,2}\ S_2$ because it is easier to read.

Figure 35 shows a graphical representation of a *4-pattern* $\langle\ S_1{=}(F_1,B),\ S_2{=}(F_3,A),\ S_3{=}(F_2,C),\ S_4{=}(F_3,B)\ \rangle$, where the states are abstractions of temporal variables F_1, F_2 and F_3 using abstraction alphabet $\Sigma = \{A,B,C\}$. The half matrix on the right represents the temporal relations between every state and the states that follow it. For example, the second state (F_3,A) *co-occurs* with the third state (F_2,C): $R_{2,3} = c$.

Definition 5. *Given an MSS* $Z = \langle\ E_1,E_2,...,E_l\ \rangle$ *and a temporal pattern* $P = (\langle S_1,...,S_k\rangle, R)$*, we say that* Z ***contains*** P*, denoted as* $\boldsymbol{P \in Z}$*, if there is an injective mapping* π *from the states*

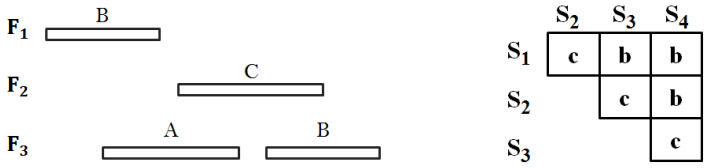


Figure 35: A temporal pattern with states $\langle (F_1,B), (F_3,A), (F_2,C), (F_3,B) \rangle$ and temporal relations $R_{1,2} = c$, $R_{1,3} = b$, $R_{1,4} = b$, $R_{2,3} = c$, $R_{2,4} = b$ and $R_{3,4} = c$.

of P to the state intervals of Z such that:

$$\forall i \in \{1,...,k\}: \; S_i.F = E_{\pi(i)}.F \; \wedge \; S_i.V = E_{\pi(i)}.V$$

$$\forall i \in \{1,...,k-1\} \wedge j \in \{i+1,...,k\}: \; R_{i,j}\left(E_{\pi(i)}\,,\, E_{\pi(j)}\right)$$

The definition says that checking whether an MSS contains a *k-pattern* requires: 1) matching all k states of the pattern and 2) checking that all $k(k-1)/2$ temporal relations are satisfied.

5.3 RECENT TEMPORAL PATTERNS

As we discussed in Section 5.1, the setting for event detection applications is that each training temporal instance x_i (e.g. an electronic health record) is associated with class label y_i at time t_i (e.g. whether or not an adverse medical event was observed). Consequently, recent measurements of the temporal variables of x_i (close to t_i) are usually more predictive than distant measurements, as was shown in [Valko and Hauskrecht, 2010] for clinical data. In the following, we define the recent temporal patterns, which are patterns that occur close to the end of the temporal instance.

Definition 6. *Given an MSS $Z = \langle E_1, E_2, ..., E_l \rangle$ and a maximum gap parameter g, we say that $E_j \in Z$ is a* ***recent state interval*** *in Z, denoted as* $\boldsymbol{r_g(E_j, Z)}$*, if* ***any*** *of the following two conditions is satisfied:*

1. $Z.end - E_j.e \leq g$.
2. $\nexists\, E_k \in Z : E_k.F = E_j.F \wedge k > j$.

The first condition is satisfied if E_j is less than g time units away from the end of the MSS ($Z.end$) and the second condition is satisfied if E_j is the most recent state interval in its variable (there is no state interval in variable $E_j.F$ that appear after E_j). Note that if $g = \infty$, any state interval of Z ($E_j \in Z$) is considered to be recent.

Definition 7. *Given an MSS $Z = \langle E_1, E_2, ..., E_l \rangle$ and a maximum gap parameter g, we say that temporal pattern $P = (\langle S_1, ..., S_k \rangle, R)$ is a* ***Recent Temporal Pattern (RTP)*** *in Z, denoted as* $\boldsymbol{R_g(P, Z)}$*, if* ***all*** *of the following conditions are satisfied:*

1. *$P \in Z$ with a mapping π from the states of P to the state intervals of Z.*
2. *S_k matches a recent state interval in Z: $r_g(E_{\pi(k)}, Z)$.*
3. *$\forall i \in \{1, ..., k-1\}$, S_i and S_{i+1} match state intervals not more than g away from each other: $E_{\pi(i+1)}.s - E_{\pi(i)}.e \leq g$.*

The definition says that in order for temporal pattern P to be an RTP in MSS Z, 1) P should be contained in Z (Definition 5), 2) the last state of P should map to a recent state interval in Z (Definition 6), and 3) any pair of consecutive states in P should map to state intervals that are "close to each other". This definition forces P to be *close to the end of Z* and to have *limited temporal extension* in the past. Note that g is a parameter that specifies the restrictiveness of the RTP definition. If $g = \infty$, any pattern $P \in Z$ would be considered to be an RTP in Z. We denote an RTP that contains k states as a ***k-RTP***.

Example 14. *Let Z be the MSS in Figure 36 and let the maximum gap parameter be $g = 3$. Temporal pattern $P = (\langle (F_4, A), (F_2, C), (F_1, B) \rangle, R_{1,2} = b, R_{1,3} = b, R_{2,3} = b)$ is an RTP in Z because $P \in Z$, $(F_1, B, 15, 18)$ is a recent state interval in Z, $(F_2, C, 8, 13)$ is "close to" $(F_1, B, 15, 18)$ $(15 - 13 \leq g)$ and $(F_4, A, 1, 5)$ is "close to" $(F_2, C, 8, 13)$ $(8 - 5 \leq g)$.*

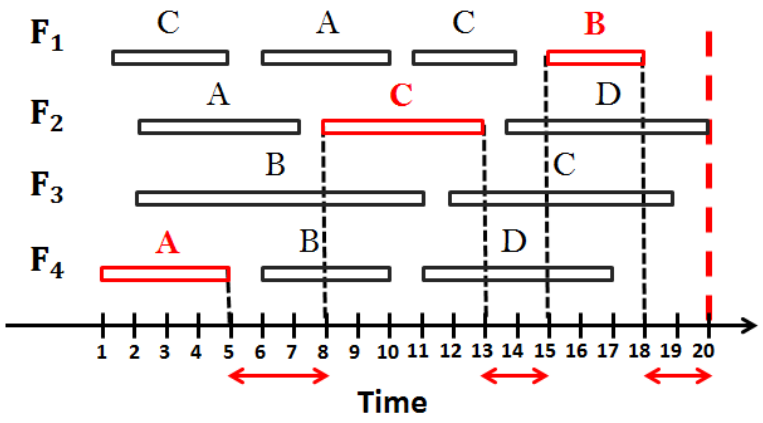


Figure 36: Temporal pattern ($\langle (F_4,A), (F_2,C), (F_1,B)\rangle$, $R_{1,2} = b$, $R_{1,3} = b$, $R_{2,3} = b$) is an RTP in this MSS if the maximum gap paramter $g \geq 3$.

Definition 8. *Given temporal patterns* $P = (\langle S_1,...,S_{k_1}\rangle, R)$ *and* $P' = (\langle S'_1,...,S'_{k_2}\rangle, R')$ *with* $k_1 < k_2$*, we say that* P *is a* ***suffix subpattern*** *of* P'*, denoted as* ***Suffix*****(P,P′)***, if:*

$$\forall i \in \{1,...,k_1\} \ \wedge \ j \in \{i+1,...,k_1\} : \ S_i = S'_{i+k2-k1} \ \wedge \ R_{i,j} = R'_{i+k2-k1,j+k2-k1}$$

That is, P consists of the last k_1 states of P' and satisfies among them the same temporal relations that are satisfied in P'. For example, pattern ($\langle (F_3,A), (F_2,C), (F_3,B)\rangle$, $R_{1,2} = c$, $R_{1,3} = b$, $R_{2,3} = c$) is a suffix subpattern of the pattern in Figure 35. When P is a suffix subpattern of P', we say that P' is a **backward-extension superpattern** of P.

Proposition 1. *Given an MSS* Z *and temporal patterns* P *and* P'*, if* P' *is an RTP in* Z *and* P *is a suffix subpattern of* P'*, then* P *is an RTP in* Z*:*

$$R_g(P',Z) \ \wedge \ Suffix(P,P') \implies R_g(P,Z)$$

The proof follows directly from the Definition of RTP.

Example 15. *Assume that* $P = (\langle S_1, S_2, S_3 \rangle, R_{1,2}, R_{1,3}, R_{2,3})$ *is an RTP in* Z. *Proposition 1 says that its suffix subpattern* ($\langle S_2, S_3 \rangle$, $R_{2,3}$) *must also be an RTP in* Z. *However, this does not imply that* ($\langle S_1, S_2 \rangle, R_{1,2}$) *must be an RTP (the second condition of Definition 7 may be violated) nor that* ($\langle S_1, S_3 \rangle, R_{1,3}$) *must be an RTP (the third condition of Definition 7 may be violated).*

Definition 9. *Given a dataset* D *of MSS and a maximum gap parameter* g, *we define the support of RTP* P *in* D *as* $\boldsymbol{RTP\text{-}sup_g(P,D)} = |\{Z_i : Z_i \in D \wedge R_g(P, Z_i)\}|$.

Given a user defined minimum support threshold σ, temporal pattern P is a **frequent RTP** in D given σ if $RTP\text{-}sup_g(P,D) \geq \sigma$.

Note that Proposition 1 implies the following property of *RTP-sup*, which we will use in our algorithm for mining frequent RTPs.

Corollary 1. *If* P *and* P' *are two temporal patterns such that* $Suffix(P, P')$, *then* $RTP\text{-}sup_g(P,D) \geq RTP\text{-}sup_g(P',D)$

5.4 MINING FREQUENT RECENT TEMPORAL PATTERNS

In this section, we present the algorithm for mining frequent RTPs. For the same reasons mentioned in Section 3.5.3, we partition the data according to the class labels and mine frequent RTPs for each class separately using a local minimum support σ_y that is related to the number of instances (MSS) from class y.

The algorithm for mining frequent RTPs for class label y takes as input the following arguments:

1. The MSS from class y: D_y.
2. The maximum gap parameter g.
3. The local minimum support threshold σ_y.

The algorithm outputs all temporal patterns that have an *RTP-sup* in D_y that is larger or equal to σ_y:

$$\{P_1, ..., P_m : RTP\text{-}sup_g(P_i, D_y) \geq \sigma_y\}$$

The algorithm explores the space of temporal patterns level by level. It first finds all frequent *1-RTPs* (recent states). Then it extends these patterns ***backward in time*** and finds frequent *2-RTPs* and so on. For each level k, the algorithm performs the following two phases to obtain the frequent *(k+1)-RTPs*:

1. **The candidate generation phase:** Generate candidate *(k+1)-patterns* by extending frequent *k-RTPs backward in time*.
2. **The counting phase:** Obtain the frequent *(k+1)-RTPs* by removing the candidates with *RTP-sup* less than σ_y.

This process repeats until no more frequent RTPs can be found.

In the following, we describe in details the candidate generation algorithm. Then we proposed techniques to improve the efficiency of candidate generation and counting.

5.4.1 Backward Candidate Generation

We generate a candidate *(k+1)-pattern* by appending a new state (*1-pattern*) to the beginning of a frequent *k-RTP*. Let us assume that we are backward extending pattern $P = (\langle S_1,...,S_k\rangle, R)$ with state S_{new} to generate candidates *(k+1)-patterns* of the form $(\langle S'_1,...,S'_{k+1}\rangle, R')$. First of all, we set $S'_1 = S_{new}$, $S'_{i+1} = S_i$ for $i \in \{1,...,k\}$ and $R'_{i+1,j+1} = R_{i,j}$ for $i \in \{1,...,k-1\} \wedge j \in \{i+1,...,k\}$. This way, we know that every candidate P' of this form is a *backward-extension superpattern* of P: $Suffix(P,P')$.

In order to fully define a candidate, we still need to specify the temporal relations between the new state S'_1 and states $S'_2,...,S'_{k+1}$, i.e., we should define $R'_{1,i}$ for $i \in \{2,...,k+1\}$. Since we have two possible temporal relations (*before* and *co-occurs*), there are 2^k possible ways to specify the missing relations, which results in 2^k different candidates. If we denote the set of all possible states by L and the set of all frequent *k-RTPs* by $\mathbf{F_k}$, generating the *(k+1)-candidates* naïvely in this fashion results in $2^k \times |L| \times |\mathbf{F_k}|$ candidate *(k+1)-patterns*.

This large number of candidates makes the mining algorithm computationally very expensive and greatly limits its scalability. Below, we describe the concept of incoherent patterns and introduce a method that generates fewer candidates without missing any real

pattern from the mining results.

5.4.2 Improving the Efficiency of Candidate Generation

Definition 10. *A temporal pattern P is **incoherent** if there does not exist any valid MSS that contains P.*

Clearly, we do not have to generate and count incoherent candidates because we know that they will have zero support in the data. We introduce the following two lemmas to avoid generating incoherent candidates when specifying the relations $R'_{1,i} : i \in \{2,...,k+1\}$ in candidates of the form $P' = (\langle S'_1,...,S'_{k+1}\rangle, R')$.

Lemma 1. *$P' = (\langle S'_1,...,S'_{k+1}\rangle, R')$ is incoherent if $\exists i \in \{2,...,k+1\} : R'_{1,i} = c$ and $S'_1.F = S'_i.F$.*

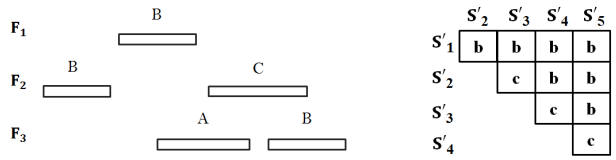
Proof. Two state intervals that belong to the same temporal variable cannot co-occur. □

Lemma 2. *$P' = (\langle S'_1,...,S'_{k+1}\rangle, R')$ is incoherent if $\exists i \in \{3,...,k+1\} : R'_{1,i} = c \wedge \exists j < i : R'_{1,j} = b$.*

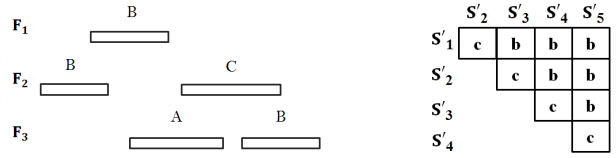
Proof. Assume that there exists an MSS $Z = \langle E_1,...,E_l\rangle$ where $P' \in Z$. Let π be the mapping from the states of P' to the state intervals of Z. The definition of temporal patterns (Definition 4) and the fact that state intervals in Z are ordered by their start values (Definition 3) implies that the matching state intervals $\langle E_{\pi(1)},...,E_{\pi(k+1)}\rangle$ are also ordered by their start times: $E_{\pi(1)}.s \leq ... \leq E_{\pi(k+1)}.s$. Hence, $E_{\pi(j)}.s \leq E_{\pi(i)}.s$ since $j < i$. We also know that $E_{\pi(1)}.e < E_{\pi(j)}.s$ because $R'_{1,j} = b$. Therefore, $E_{\pi(1)}.e < E_{\pi(i)}.s$. However, since $R'_{1,i} = c$, then $E_{\pi(1)}.e \geq E_{\pi(i)}.s$, which is a contradiction. Therefore, there is no MSS that contains P'. □

Example 16. *Assume we want to extend $P = (\ \langle S_1 = (F_1, B),\ S_2 = (F_3, A),\ S_3 = (F_2, C),\ S_4 = (F_3, B)\rangle,\ R_{1,2} = c,\ R_{1,3} = b,\ R_{1,4} = b,\ R_{2,3} = c,\ R_{2,4} = b,\ R_{3,4} = c\ )$ in Figure 35 with state $S_{new} = (F_2, B)$ to generate candidates of the form $(\langle S'_1 = (F_2, B),\ S'_2 = (F_1, B),\ S'_3 = (F_3, A),\ S'_4 = (F_2, C),\ S'_5 = (F_3, B)\rangle,\ R')$. We have to specify relations $R'_{1,i} : i \in \{2,...,k+1\}$. $R'_{1,2}$ is allowed to be either before ($R'_{1,2} = b$) or co-occurs ($R'_{1,2} = c$). If $R'_{1,2} = b$, then all the following relations must be before according to Lemma 2, resulting in the candidate shown in Figure 37:a. If $R'_{1,2} = c$, then $R'_{1,3}$ is allowed to be either before ($R'_{1,3} = b$) or co-occurs ($R'_{1,3} = c$), resulting in the candidates shown in Figure 37:b and Figure 37:c, respectively. Now, according to Lemma*

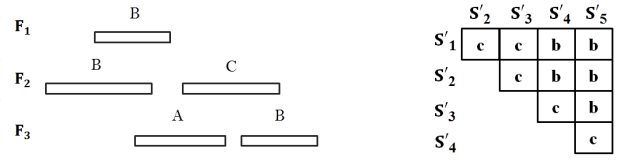
1, $R'_{1,4} \neq c$ *because both* S'_1 *and* S'_4 *belong to the same temporal variable* (F_2). *As we see, we reduce the number of candidates that result from adding* (F_2, B) *to 4-RTP* P *from* $2^4 = 16$ *in the naïve way to only 3.*



(a)



(b)



(c)

Figure 37: The coherent candidates that result from extending the temporal pattern in Figure 35 backward in time with state (F_2, B).

Theorem 1. *There are at most* $k+1$ *coherent candidates that result from backward extending a single k-RTP with a new state.*

Proof. We know that every candidate $P' = (\langle S'_1, ..., S'_{k+1} \rangle, R')$ corresponds to a specific assignment of $R'_{1,i} \in \{b, c\}$ for $i \in \{2, ...k+1\}$. When we assign the temporal relations, once a relation becomes *before*, all the following relations have to be *before* as well according to Lemma 2. We can see that the relations can be *co-occurs* in the beginning of the pattern, but once we have a *before* relation at point $q \in \{2, ..., k+1\}$ in the pattern, all subsequent relations $(i > q)$

should be *before* as well:

$$R'_{1,i} = c : i \in \{2,...,q-1\}; \quad R'_{1,i} = b : i \in \{q,...,k+1\}$$

Therefore, the total number of coherent candidates cannot be more than $k+1$, which is the total number of different combinations of consecutive *co-occurs* relations followed by consecutive *before* relations. □

In some cases, the number of coherent candidates is less than $k+1$. Assume that there are some states in P' that belong to the same variable as state S'_1. Let S'_j be the first such state ($j \leq k+1$). According to Lemma 1, $R'_{1,j} \neq c$. In this case, the number of coherent candidates is $j-1 < k+1$.

Algorithm 1 illustrates how to extend a *k-RTP* P with a new state S_{new} to generate coherent candidates (without violating Lemmas 1 and 2).

Algorithm 1: Extend backward a *k-RTP* P with a state S_{new}.

Input: A *k-RTP*: $P = (\langle S_1,...,S_k\rangle, R)$; a new state: S_{new}

Output: Coherent candidates: C

```
1  S'_1 = S_new; S'_{i+1} = S_i : i ∈ {1,...,k};
2  R'_{i+1,j+1} = R_{i,j} : i ∈ {1,...,k-1}, j ∈ {i+1,...,k};
3  R'_{1,i} = b : i ∈ {2,...,k+1};   P' = (⟨S'_1,...,S'_{k+1}⟩, R');
4  C = {P'};
5  for i=2 to k+1 do
6      if (S'_1.F = S'_i.F) then
7          break;
8      else
9          R'_{1,i} = c;   P' = (⟨S'_1,...,S'_{k+1}⟩, R');
10         C = C ∪ {P'};
11     end
12 end
13 return C
```

Corollary 2. *Let L denote the set of all possible states and let $\mathbf{F_k}$ denote the set of all frequent k-RTPs. The number of coherent (k+1)-candidates is always less or equal to* $(k+1) \times |L| \times |\mathbf{F_k}|$.

5.4.3 Improving the Efficiency of Counting

Even after eliminating incoherent candidates, the mining algorithm is still computationally expensive because for every candidate, we need to scan the *entire database* in the counting phase to compute its *RTP-sup*. The question we try to answer in this section is whether we can omit portions of the database that are guaranteed not to contain the candidate we want to count. The proposed solution is inspired by [Zaki, 2000] that introduced the *vertical format* for itemset mining (see Section 2.2.3) and later applied it for sequential pattern mining [Zaki, 2001].

Let us associate every frequent RTP P with a list of identifiers for all MSS in D_y that have P as an RTP (Definition 7):

$$P.RTP\text{-}list = \langle i_1, i_2, ..., i_n \rangle : Z_{i_j} \in D_y \wedge R_g(P, Z_{i_j})$$

Clearly, $RTP\text{-}sup_g(P, D_y) = |P.RTP\text{-}list|$.

Let us also associate every state S with a list of identifiers for all MSS that contain S (Definition 5):

$$S.list = \langle q_1, q_2, ..., q_m \rangle : Z_{q_j} \in D_y \wedge S \in Z_{q_j}$$

Now, when we generate candidate P' by backward extending RTP P with state S, we define the potential list (***p-RTP-list***) of P' as follows:

$$P'.p\text{-}RTP\text{-}list = P.RTP\text{-}list \cap S.list$$

Proposition 2. *Let P' be a backward-extension superpattern of RTP P with state S: $P'.RTP\text{-}list \subseteq P'.p\text{-}RTP\text{-}list$*

Proof. Assume Z_i is an MSS such that $R_g(P', Z_i)$. By definition, $i \in P'.RTP\text{-}list$. We know that $R_g(P', Z_i) \Longrightarrow P' \in Z_i \Longrightarrow S \in Z_i \Longrightarrow i \in S.list$. Also, we know that $Suffix(P, P')$ (Definition 8) $\Longrightarrow R_g(P, Z_i)$ (Proposition 1) $\Longrightarrow i \in P.RTP\text{-}list$. Therefore, $i \in P.RTP\text{-}list \cap S.list = P'.p\text{-}RTP\text{-}list$ □

Putting it all together, we compute the $RTP\text{-}lists$ in the *counting phase* (based on the true matches) and the $p\text{-}RTP\text{-}lists$ in the *candidate generation phase*. The key idea is that when we count candidate P', we only need to check the instances in its $p\text{-}RTP\text{-}list$ because according to Proposition 2: $i \notin P'.p\text{-}RTP\text{-}list \Longrightarrow i \notin P'.RTP\text{-}list \Longrightarrow P'$ is not an RTP in Z_i. This offers a lot of computational savings because the $p\text{-}RTP\text{-}lists$ get smaller as the size of the patterns increases, making the counting phase much faster.

Algorithm 2 outlines the candidate generation. Line 4 generates coherent candidates using Algorithm 1. Line 6 computes the $p\text{-}RTP\text{-}list$ for each candidate. Note that the cost of the intersection is *linear* because the lists are always sorted according to the order of the instances in the data. Line 7 applies an *additional pruning* to remove candidates that are guaranteed not to be frequent according to the following implication of Proposition 2:

$$|P'.p\text{-}RTP\text{-}list| < \sigma_y \Longrightarrow |P'.RTP\text{-}list| = RTP\text{-}sup_g(P, D_y) < \sigma_y$$

5.5 MINING MINIMAL PREDICTIVE RECENT TEMPORAL PATTERNS

Although the RTP framework focuses the search on temporal patterns that are potentially important for predicting the class variable, not all frequent RTPs are important for classification. Besides, many RTPs may be spurious (see Section 3.4) as we illustrate in the following example.

Example 17. *Assume that having elevated creatinine level (creatinine=High) is an important indicator of renal failure. If we denote this pattern by P, we expect conf(P ⇒ renal-failure) to be much higher than the renal-failure prior in the entire population of patients.*

Algorithm 2: A high-level description of candidate generation.

Input: All frequent *k-RTPs*: $\mathbf{F_k}$; all frequent states: L

Output: Candidate *(k+1)-patterns*: $Cand$, with their *p-RTP-lists*

```
1  Cand = Φ;
2  foreach P ∈ F_k do
3      foreach S ∈ L do
4          C = extend_backward(P, S);    (Algorithm 1)
5          for q = 1 to | C | do
6              C[q].p-RTP-list = P.RTP-list ∩ S.list;
7              if ( | C[q].p-RTP-list | ≥ σ_y ) then
8                  Cand = Cand ∪ {C[q]};
9              end
10         end
11     end
12 end
13 return Cand
```

Now consider pattern P' that extends P backward with a state indicating a normal value for white blood cell counts: P': WBC=Normal before creatinine=High. Assume that observing P' does not change our belief about the presence of renal failure compared to observing P. As we discussed in Section 3.4, conf($P' \Rightarrow$ renal-failure) $\approx$ conf($P \Rightarrow$ renal-failure). Intuitively, the instances covered by P' can be seen as a random sample of the instances covered by P. So if the proportion of renal failure for P is relatively high, we expect the proportion of renal failure for P' to be high as well. The problem is that if we evaluate P' by itself, we may falsely think that it is an important pattern for predicting renal failure, where in fact this happens only because P' contains the real predictive pattern P.

In general, spurious RTPs are formed by adding irrelevant states to other simpler predictive RTPs. Having spurious RTPs in the result is undesirable because they overwhelm the user and prevent him/her from understanding the important patterns in data. In order

to filter out such spurious patterns, we extend the minimal predictive patterns framework (Section 3.5.2) to the temporal domain.

Definition 11. *A temporal pattern P is a **Minimal Predictive Recent Temporal Pattern** (**MPRTP**) with respect to class label y if P predicts y significantly better than **all** of its suffix subpatterns.*

$$\forall S \text{ such that } Suffix(S,P):\ BS(P \Rightarrow y, G_S) \geq \delta$$

Where BS is the Bayesian score we defined in Section 3.5.1.2, G_S is the group of MSS in the data where S is an RTP and δ is a user specified significance parameter.

The algorithm in Section 5.4 describes how to mine all frequent RTPs from data D_y. In order to mine MPRTPs, the algorithm requires another input: $D_{\neg y}$, the MSS in the data that do not belong to class y. Mining MPRTPs is integrated with frequent RTP mining using an algorithm similar to the one described in Section 3.5.3 for mining MPPs. The algorithm utilizes the *predictiveness* of RTPs to prune the search space using a lossless pruning technique and a lossy pruning technique.

lossless pruning: This technique is similar to the lossless pruning used for MPP mining (Section 3.5.4.1). The idea is to prune a frequent RTP P if we guarantee that none of its backward-extension superpatterns is going to be an MPRTP. We know that for any backward-extension superpattern P', the following holds according to Corollary 1:

$$RTP\text{-}sup_g(P',D_y) \leq RTP\text{-}sup_g(P,D_y)\ \wedge\ RTP\text{-}sup_g(P',D_{\neg y}) \leq RTP\text{-}sup_g(P,D_{\neg y})$$

We now define the optimal backward-extension superpattern of P with respect to class y, denoted as P^*, to be a hypothetical temporal pattern that is an RTP in all instances from y, but not in any instance from from the other classes:

$$RTP\text{-}sup_g(P^*,D_y) = RTP\text{-}sup_g(P,D_y)\ \wedge\ RTP\text{-}sup_g(P^*,D_{\neg y}) = 0$$

P^* is the best possible backward-extension superpattern for predicting y that P can generate. Now, we *safely* prune P if P^* does not satisfy the MPRTP definition.

lossy pruning: This technique is similar to the lossy pruning used for MPP mining (Section 3.5.4.2). The idea is that if we are mining MPRTPs for class y, we prune RTP P if we have evidence that the underlying probability of y in G_P (the group of MSS in D_y where

P is an RTP) is lower than the probability of y in the entire data. To decide whether this is the case, we apply our Bayesian score to evaluate rule $P \Rightarrow y$ compared to G_ϕ and we prune P if model M_l is the most likely model (see Section 3.5.1.2).

The rationale behind this heuristic is that if the probability of y in the MSS covered by P is low, we also expect the probability of y in the MSS covered by its backward-extension superpattern P' to be low as well. Thus, P' is unlikely to be an MPRTP. Note that this heuristic is lossy in the sense that it speeds up the mining, but at the risk of missing some MPRTPs.

5.6 LEARNING THE EVENT DETECTION MODEL

In this section, we summarize our approach for learning classification models for event detection problems. Given a training dataset $\{<\boldsymbol{x_i}, y_i>\}_{i=1}^{n}$, where x_i is a multivariate time series instance up to time t_i and y_i is a class label at t_i, we apply the following steps:

1. Convert every instance x_i to an MSS Z_i using temporal abstraction.
2. Mine the frequent RTPs or MPRTPs from the MSS of each class label separately and combine the class-specific patterns to obtain the final result Ω.
3. Convert every MSS Z_i into a binary vector x'_i of size equal to $|\Omega|$, where $x'_{i,j}$ corresponds to a specific pattern $P_j \in \Omega$ and its value is 1 if $R_g(P_j, Z_i)$; and 0 otherwise.
4. Learn the classification model on the transformed binary representation of the training data $\{<\boldsymbol{x'_i}, y_i>\}_{i=1}^{n}$.

5.7 EXPERIMENTAL EVALUATION

5.7.1 Temporal Datasets

In our experiments, we evaluate our RTP and MPRTP frameworks first on a synthetic dataset and then on two real-world EHR dataset.

5.7.1.1 Synthetic Dataset This data consist of *multivariate symbolic sequences* (*time-point* states). Each instance has 6 temporal variables (F_1 to F_6) and each temporal variable is a sequence of exactly 4 states. All states are *regularly sampled in time* and belong to alphabet $\Sigma = \{A, B, C\}$ (see Figure 38).

We define two class labels, y_1 and y_2, and generate 250 instances from each class as follows:

- Step I: We randomly generate the instances of both classes by sampling each state in each variable from alphabet $\Sigma = \{A, B, C\}$ according to a multinomial distribution with probabilities: $Pr(A) = 0.25$, $Pr(B) = 0.25$ and $Pr(C) = 0.5$.
- Step II: For the instances from class y_1, we inject RTP P_1: $< F_1 = B,\ F_1 = A >$ (the last two states of F_1 are B followed by A as in Figure 38:a) with probability 0.6. For the instances from class y_2, we inject RTP P_2: $< F_2 = C,\ F_3 = C,\ F_2 = C >$ (the second state of F_2 is C, the third state of F_3 is C and the fourth state of F_2 is C as in Figure 38:b) with probability 0.45.

Note that the probability of observing RTP P_1 (by chance) in an instance from class y_2 is 0.0625. The reason is that the probability that third state of F_1 is B and the fourth state of F_1 is A in class y_2 is $0.25 \times 0.25 = 0.0625$. On the other hand, the probability of observing RTP P_1 in an instance from class y_1 is 0.625. The reason is that P_1 is injected in 60% of the instances of y_1 and can occur by chance with probability 0.0625 in the remaining 40% of the instances ($0.6 + 0.4 \times 0.0625 = 0.625$). Using a similar reasoning, we can see that the probability of observing RTP P_2 in an instance from class y_1 is 0.125 and in an instance from class y_2 is 0.519. Both of these RTPs are important for discriminating the two classes and we would like our mining algorithm to be able to recover them.

5.7.1.2 HIT Dataset This data are acquired from a database that contains electronic health records of post cardiac surgical patients [Hauskrecht et al., 2010]. Our task is to learn a classifier that can detect the onset of *Heparin Induced Thrombocytopenia* (*HIT*), which is a pro-thrombotic disorder induced by heparin exposure with subsequent thrombocytopenia (low platelet in the blood) and associated thrombosis (blood clot). HIT is a life-threatening

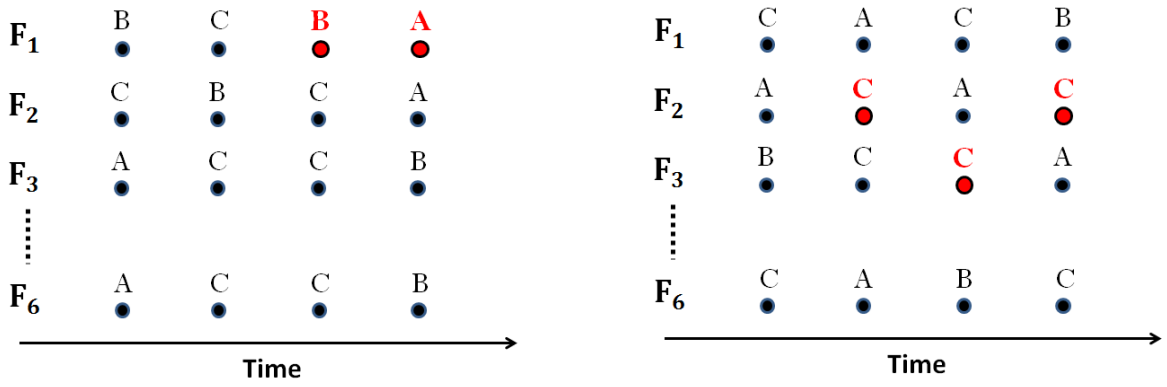


Figure 38: The synthetic temporal data: on the left we show RTP $< F_1 = B,\ F_1 = A >$ that is injected in the instances of class y_1 and on the right we show RTP $< F_2 = C,\ F_3 = C,\ F_2 = C >$ that is injected in the instances of class y_2.

condition if it is not detected and managed properly. Hence, it is very important to detect its onset.

We select 220 patients who were considered by physicians to be at risk of HIT and 220 patients without the risk of HIT. Patients who are at risk of HIT were selected using information about the *Heparin Platelet Factor 4 antibody* (*HPF4*) test, which is ordered for a patient when the physician suspects that he is developing HIT. Therefore, an HPF4 test order is a good surrogate of the *HIT-risk* label. The positive instances (*HIT-risk*) include clinical information up to the time HFP4 was first ordered. The negative instances (*no HIT-risk*) were selected randomly from the remaining patients and they include clinical information up to some randomly selected time point in the patient's record.

For each instance, we consider the following 5 clinical variables: platelet counts (PLT), activated partial thromboplastin time (APTT), white blood cell counts (WBC), hemoglobin (Hgb) and heparin orders. PLT, APTT, WBC and Hgb are numeric time series, so we convert them into time-interval sequences using both *trend abstractions* and *value abstractions* (Section 5.2.1). Heparin orders are already in an interval-based format that specify the

time period during which the patient was taking heparin. For this variable, we simply use abstractions that indicate whether the patient is on heparin.

5.7.1.3 Diabetes Dataset This data consist of 13,558 electronic health records of adult diabetic patients (both type I and type II diabetes). Our task is to learn classifiers that can detect various types of disorders that are frequently associated with diabetes.

Each patient's record consists of time series of 19 different lab values, including blood glucose, creatinine, glycosylated hemoglobin, blood urea nitrogen, liver function tests, cholesterol, etc. In addition, we have access to time series of ICD-9 diagnosis codes reflecting the diagnoses made for the patient over time. Overall, the database contains 602 different ICD-9 codes. These codes were grouped by our medical expert into the following eight major diagnosis categories (diseases):

- Cardiovascular disease (**CARDI**).
- Renal disease (**RENAL**).
- Peripheral vascular disease (**PERIP**).
- Neurological disease (**NEURO**).
- Metabolic disease (**METAB**).
- Inflammatory (infectious) disease (**INFLM**).
- Ocular (ophthalmologic) disease (**OCULR**).
- Cerebrovascular disease (**CEREB**).

Our objective is to learn models that are able to accurately diagnose each of these major diseases. For each disease, we divide the data into positive instances and negative instances as follows:

- The positives are records of patients with the target disease and they include clinical information up to the time the disease was *first* diagnosed.
- The negatives are records of patients without the target disease and they include clinical information up to a randomly selected time point in the patient's record.

To avoid having uninformative data, we discard instances that contain less than 10 lab measurements or that span less than 3 months (short instances). *We choose the same number of controls as the number of cases for each category to make the datasets balanced.*

For each instance, we consider both the laboratory tests and the disease categories. Note that the diagnosis of one or more disease categories may be predictive of the (first) occurrence of another disease, so it is important to include them as features. Laboratory tests are numeric time series, so we convert them into time-interval sequences using *value abstractions* (see Section 5.2.1). Disease categories, when used as features, are represented as intervals that start at the time of the diagnosis and extend until the end of the record. For these variables, we simply use abstractions that indicate whether the patient has been diagnoses with the disease.

5.7.1.4 Datasets Summary Table 10 summaries the three temporal datasets we use in our experimental evaluation. For each dataset, we show its type (whether the states are time-points or time-intervals), the number of data instances it contains, the number of temporal variables per instance, the alphabet size (the number of permitted abstractions) and the total number of states in the data.

For the synthetic dataset, we have 500 instances (250 instances from each class) and 6 temporal variables per instance. The alphabet size is 3 because $\Sigma = \{A, B, C\}$. Since there are 4 states per variable, the number of states per instance is $6 \times 4 = 24$ and the total number of states in the data is $24 \times 500 = 12,000$.

For the HIT dataset, we have 440 instances (220 *HIT-risk* instances and 220 *no HIT-risk* instances). We use both trend abstractions (decreasing, steady and increasing) and value abstractions (very low, low, normal, high and very high) for the laboratory variables and one abstraction for the heparin variable. Hence, the alphabet size is 9. Since we consider two types of temporal abstractions for each of the 4 laboratory variables (PLT, APTT, WBC and Hgb) and one abstraction for the heparin variable, we have a total of 9 temporal variables per instance.

For the diabetes datasets, we define a different dataset for each of the 8 major diagnosis diseases (see Section 5.7.1.3). For all of these datasets, we use value abstractions (very low,

low, normal, high and very high) for the laboratory variables and one abstraction for the disease categories. Hence, the alphabet size is 6. Since we have 19 laboratory variables and 7 disease variables (the 8 major diseases minus the one we are predicting), we have a total of 26 temporal variables per instance.

Dataset	Type	# Instances	# Variables	Alphabet Size	# States
Synthetic	Time-point	500	6	3	12,000
HIT	Time-interval	440	9	9	9,770
Diabetes-CARDI	Time-interval	5,486	26	6	235,990
Diabetes-RENAL	Time-interval	6,710	26	6	327,957
Diabetes-PERIP	Time-interval	6,740	26	6	325,872
Diabetes-NEURO	Time-interval	4,386	26	6	240,572
Diabetes-METAB	Time-interval	1,936	26	6	118,378
Diabetes-INFLM	Time-interval	4,788	26	6	264,541
Diabetes-OCULR	Time-interval	4,490	26	6	227,708
Diabetes-CEREB	Time-interval	5,648	26	6	319,695

Table 10: Temporal datasets characteristics.

5.7.2 Classification

In this section, we test the performance of our RTP mining and MPRTP mining methods for the event detection task.

5.7.2.1 Compared Methods We compare the classification performance of the following feature construction methods:

1. ***Last-abs:*** The features are the most recent abstractions of each clinical variable. For example, the most recent trend abstraction for platelet counts is "*decreasing*", the most recent value abstraction for platelet counts is "*low*", and so on.
2. ***TP:*** The features correspond to all frequent temporal patterns.

$$\{P_1,...,P_m : sup(P_i,D_y) \geq \sigma_y\} \quad where \quad sup(P,D_y) = |\{Z_i : Z_i \in D_y \wedge P \in Z_i\}|$$

3. ***TP-IG:*** The features correspond to the top k frequent temporal patterns, where the patterns are ranked according to Information Gain (IG).
4. ***RTP:*** The features correspond to all frequent RTPs.

$$\{P_1,...,P_k : RTP\text{-}sup_g(P_i,D_y) \geq \sigma_y\} \quad where \quad RTP\text{-}sup_g(P,D_y) = |\{Z_i : Z_i \in D_y \wedge R_g(P_j,Z_i)\}|$$

5. ***RTP-IG:*** The features correspond to the top k frequent RTPs, where the patterns are ranked according to IG.
6. ***MPRTP:*** The features correspond to the top k frequent RTPs, where only the patterns that satisfy the MPRTP definition (Definition 11) are retained and they are ranked according to the Bayesian score (see Section 3.5.1.2).

The first method (*Last-abs*) is *atemporal* and only considers the most recent abstractions for defining the classification features (a static transformation). On the other hand, methods (2-6) use temporal patterns (built using temporal abstractions and temporal relations) as their features (a dynamic transformation).

When defining the binary representation of an instance (MSS) Z_i for methods *TP* and *TP-IG*, the feature value is set to one if the corresponding temporal pattern occurs anywhere in the instance (Definition 5), and is set to zero otherwise:

$$Z_i \rightarrow x'_i \text{ where } x'_{i,j} = 1 \text{ if } P_j \in Z_i \text{ and } x'_{i,j} = 0 \text{ otherwise.}$$

When defining the binary representation of an instance Z_i for methods *RTP*, *RTP-IG* and *MPRTP*, the feature value is set to one if the corresponding temporal pattern occurs recently in the instance (Definition 7), and is set to zero otherwise (see Section 5.6):

$$Z_i \rightarrow x'_i \text{ where } x'_{i,j} = 1 \text{ if } R_g(P_j,Z_i) \text{ and } x'_{i,j} = 0 \text{ otherwise.}$$

It is important to note that although patterns generated by *TP* subsume the ones generated by *RTP* (by definition, every frequent RTP is also a frequent temporal pattern), the induced binary features are often different. For example, a temporal pattern that is very discriminative when observed at the end of an instance may become less discriminative when observed in the middle of an instance.

We use methods *TP-IG*, *RTP-IG* and *MPRTP* in the evaluation because we want to compare the ability of *TP* and *RTP* in representing the classifier using only a limited number of temporal patterns (a sparse classifier). In addition, we want to compare using a univariate scoring versus our *MPRTP* approach for selecting RTPs that are used for classification.

We judged the quality of the different feature representations in terms of their induced classification performance. More specifically, we use the features extracted by each method to build a **linear SVM** classifier and evaluate its performance using the area under the ROC curve (AUC) and the classification accuracy. We did not compare against other time series classification methods because most methods [Blasiak and Rangwala, 2011, Batal and Hauskrecht, 2009, Weng and Shen, 2008, Vail et al., 2007, Xi et al., 2006] cannot be directly applied on multivariate irregularly sampled time series data as our EHR data.

All classification results are reported using averages obtained via *10-fold cross-validation*, where the same train/test splits are used for all compared methods.

5.7.2.2 Results on Synthetic Data For all temporal pattern mining methods (*TP*, *TP-IG*, *RTP*, *RTP-IG* and *MPRTP*), we set the local minimum supports (σ_y) to 10% of the number of instances in the class. For *RTP*, *RTP-IG* and *MPRTP*, we set the maximum gap parameter (see Definition 7) to 1 time unit, which means that we do not allow gaps between consecutive states of an RTP[5]. For *TP-IG*, *RTP-IG* and *MPRTP*, we select the top 10 patterns to be used for classification.

Table 11 shows the AUC and the classification accuracy on the synthetic dataset. We show the best performing method in boldface and we show all methods that are statistically significantly inferior to it in grey. For the statistical test, we apply paired t tests (see chapter

[5]Remember that the synthetic data is regularly sampled in time.

5 in [Mitchell, 1997]) with the commonly used 0.05 significance level[6].

We can see that *RTP* and *MPRTP* are the best performing methods in terms of AUC. Also, the AUC of *MPRTP* is much higher than that of *RTP-IG*. The reason is that *RTP-IG* evaluates the RTPs individually, hence most of the selected RTPs are spurious and do not improve to the classification performance. For example, a spurious RTP like $< F_3 = C,\ F_1 = B,\ F_1 = A >$, which is a backward-extension superpattern of $< F_1 = B,\ F_1 = A >$ (Figure 38:a), have a higher information gain score than $< F_2 = C,\ F_3 = C,\ F_2 = C >$ (Figure 38:b), which is much more important for classification.

	Last-abs	*TP*	*TP-IG*	*RTP*	*RTP-IG*	*MPRTP*
AUC	82.51	85.93	82.44	87.06	81.34	**88.07**
Accuracy	73.8	75.8	77.8	79.3	80.4	**80.6**

Table 11: **Synthetic dataset**: The area under ROC (AUC) and the classification accuracy of the compared feature representation methods (Section 5.7.2.1). The best performing method is shown in bold and all methods that are statistically inferior to it are shown in grey. SVM is used for classification.

5.7.2.3 Results on HIT Data For all temporal pattern mining methods (*TP*, *TP-IG*, *RTP*, *RTP-IG* and *MPRTP*), we set the local minimum supports (σ_y) to 10% of the number of instances in the class. For *RTP*, *RTP-IG* and *MPRTP*, we set the maximum gap parameter (see Definition 7) to 2 days. For *TP-IG*, *RTP-IG* and *MPRTP*, we select the top 50 patterns to be used for classification.

Table 12 shows the AUC and the classification accuracy on the HIT dataset. We can see that features based on temporal patterns (*TP*, *RTP* and *MPRTP*) are beneficial for the classification task, since they outperform features based on only most recent abstractions

[6]We apply statistical significance testing with k-fold cross validation. It is important to note that in this setting, the testing sets are independent of each other, but the training sets are not independent. Hence, the statistical models do not perfectly fit the iid assumption. Nevertheless, the significance results are still of great help in interpreting experimental comparison of learning methods [Mitchell, 1997].

(*last-abs*). For the temporal pattern mining methods, *RTP* and *MPRTP* are the best performing methods.

	Last-abs	*TP*	*TP-IG*	*RTP*	*RTP-IG*	*MPRTP*
AUC	87.18	90.87	87.79	**91.99**	88.58	91.57
Accuracy	79.52	80.62	82	**84.3**	82.46	83.61

Table 12: **HIT dataset**: The area under ROC (AUC) and the classification accuracy of the compared feature representation methods (Section 5.7.2.1). The best performing method is shown in bold and all methods that are statistically inferior to it are shown in grey. SVM is used for classification.

5.7.2.4 Results on Diabetes Data For all temporal pattern mining methods (*TP*, *TP-IG*, *RTP*, *RTP-IG* and *MPRTP*), we set the local minimum supports (σ_y) to 15% of the number of instances in the class. For *RTP*, *RTP-IG* and *MPRTP*, we set the maximum gap parameter (see Definition 7) to 6 months[7]. For *TP-IG*, *RTP-IG* and *MPRTP*, we select the top 50 patterns to be used for classification.

Table 13 and Table 14 show the AUC and the classification accuracy for each classification task (major disease). We can see that for most classification tasks, *RTP* is the best performing method. We can see that although *MPRTP* does not perform as well as *RTP*, it mostly outperforms *RTP-IG* (see for example the performance on the *NEURO* dataset in Table 13 and Table 14).

[7]Note that the diabetes data are outpatient data and have a much coarser time granularity than the inpatient HIT data.

	Last-abs	*TP*	*TP-IG*	*RTP*	*RTP-IG*	*MPRTP*
CARDI	77.52	80.03	77.28	**80.04**	78.74	79.43
RENAL	83.28	84.97	73.38	**86.27**	83.65	84.41
PERIP	75.11	76	73.53	**77.95**	75.72	75.82
NEURO	72.2	74.46	72.03	**76.23**	71.89	74.33
METAB	80.8	**83**	80.17	81.65	80.76	82.59
INFLM	72.21	73.2	70.93	**74.49**	72.52	73.19
OCULR	73.71	**76.65**	74.92	75.52	74.74	75.23
CEREB	72.69	75.22	72.53	**76.3**	73.34	73.66

Table 13: **Diabetes dataset**: The area under ROC of the compared feature representation methods (Section 5.7.2.1) for the eight major diabetes diseases (Section 5.7.1.3). The best performing method is shown in bold and all methods that are statistically inferior to it are shown in grey. SVM is used for classification.

	Last-abs	*TP*	*TP-IG*	*RTP*	*RTP-IG*	*MPRTP*
CARDI	69.23	**72.35**	69.14	71.54	70.61	71.25
RENAL	74.56	76.99	71.89	**77.91**	76.87	76.9
PERIP	67.22	68.4	66.47	**69.58**	68.55	68.52
NEURO	64.5	67.49	65.21	**68.49**	65.98	67.26
METAB	71.62	**74.12**	72.21	73.09	72.21	72.83
INFLM	65.31	66.06	63.79	**67.19**	65.48	66.58
OCULR	67.04	**69.67**	67.44	68.15	67.57	67.46
CEREB	65.64	67.78	65.49	**68.49**	66.08	65.65

Table 14: **Diabetes dataset**: The classification accuracy of the compared feature representation methods (Section 5.7.2.1) for the eight major diabetes diseases (Section 5.7.1.3). The best performing method is shown in bold and all methods that are statistically inferior to it are shown in grey. SVM is used for classification.

5.7.3 Knowledge Discovery

In this section, we test the ability of *MPRTP* for mining concise predictive and non-spurious RTPs.

5.7.3.1 Results on Synthetic Data Table 16 shows the top 3 MPRTPs according to the Bayesian score on the synthetic data. Patterns P_1 and P_3 recover the two patterns we injected in the instances of class y_1 and class y_2 (see Figure 38). Pattern P_2 is a suffix subpattern of P_1. As we discuss in Section 5.7.2.2, if we use a univariate evaluation measure (such as IG) instead of *MPRTP*, many spurious backward-extension superpatterns of $< F_1 = B,\ F_1 = A >$ will be ranked higher than $< F_2 = C,\ F_3 = C,\ F_2 = C >$.

MPRTP	*Precision*	*Recall*
P_1: $< F_1 = B,\ F_1 = A > \ \Rightarrow\ y_1$	93.18	65.6
P_2: $< F_1 = A > \ \Rightarrow\ y_1$	74.48	71.2
P_3: $< F_2 = C,\ F_3 = C,\ F_2 = C > \ \Rightarrow\ y_2$	76.54	49.6

Table 15: **Synthetic dataset**: The top 3 MPRTPs with their precision and recall.

5.7.3.2 Results on HIT Data Table 16 shows the top 5 MPRTPs according to the Bayesian score on the HIT data. Patterns P_1, P_2, P_3 and P_4 describe the main patterns used to detect HIT and are in agreement with the current HIT detection guidelines [Warkentin, 2000]. P_5 relates the risk of HIT with an increasing trend of APTT (activated partial thromboplastin time). This relation is not obvious from the HIT detection guidelines. However it has been recently discussed in the literature [Pendelton et al., 2006]. Hence this pattern requires further investigation.

MPRTP	*Precision*	*Recall*
P_1: *PLT=L* $\Rightarrow$ *HIT-risk*	78.3	84.79
P_2: *PLT=VL* $\Rightarrow$ *HIT-risk*	89.31	65.44
P_3: *PLT=L before PLT=VL* $\Rightarrow$ *HIT-risk*	91.13	52.07
P_4: *PLT=D co-occurs PLT=L* $\Rightarrow$ *HIT-risk*	86.33	55.3
P_5: *APTT=I before PLT=L* $\Rightarrow$ *HIT-risk*	88.24	41.47

Table 16: **HIT dataset**: The top 5 MPRTPs with their precision and recall. Abbreviations: *PLT*: platelet count; *APTT*: activated partial thromboplastin time. Trend abstractions: *PLT=D*: decreasing trend in *PLT*; *APTT=I*: increasing trend in *APTT*. Value abstractions: *PLT=VL* (Very Low): $<76\times10^9$ per liter; *PLT=L* (Low): $<118\times10^9$ per liter.

5.7.3.3 Results on Diabetes Data Table 17 shows some of the top MPRTPs[8] on the diabetes data. Patterns P_1, P_2 and P_3 are predicting renal (kidney) disease. These patterns relate the risk of renal problems with very high values of the BUN test (P_1), an increase in creatinine levels from normal to high (P_2), and high values of BUN co-occurring with high values of creatinine (P_3). P_4 shows that an increase in glucose levels from high to very high may indicate a metabolic disease. Finally, P_5 shows that patients who were previously diagnosed with cardiovascular disease and exhibit an increase in glucose levels are prone to develop a cerebrovascular disease. These patterns, extracted automatically from data without prior clinical knowledge, are in accordance with the medical diagnosis guidelines.

[8]Most of the highest scores MPRTPs are predicting the RENAL category because it is the easiest prediction task. So to diversify the patterns, we show the top 3 predictive MPRTPs for RENAL and the top 2 MPRTPs for other categories.

MPRTP	*Precision*	*Recall*
P_1: *BUN=VH* ⇒ *Dx=RENAL*	0.97	0.17
P_2: *Creat=N before Creat=H* ⇒ *Dx=RENAL*	0.96	0.21
P_3: *BUN=H co-occurs Creat=H* ⇒ *Dx=RENAL*	0.95	0.21
P_4: *Gluc=H before Gluc=VH* ⇒ *Dx=METAB*	0.79	0.24
P_5: *Dx=CARDI co-occurs* (*Gluc=N before Gluc=H*) ⇒ *Dx=CEREB*	0.71	0.22

Table 17: **Diabetes dataset**: The top MPRTPs with their precision and recall. Abbreviations: *Dx*: diagnosis code (one of the 8 major categories described in Section 5.7.1.3); *BUN*: Blood Urea Nitrogen; *Creat*: creatinine; *Gluc*: blood glucose. Value abstractions: *BUN=VH*: >49 mg/dl; *BUN=H*: >34 mg/dl; *Creat=H*: >1.8 mg/dl; *Creat=N*: [0.8-1.8] mg/dl; *Gluc=VH*: >243 mg/dl; *Gluc=H*: >191 mg/dl.

5.7.4 Mining Efficiency

In this section, we study the efficiency of different temporal pattern mining methods.

5.7.4.1 Compared Methods We compare the running time of the following methods:

1. ***TP_Apriori:*** Mine frequent temporal patterns by extending the Apriori algorithm [Agrawal and Srikant, 1994, Agrawal and Srikant, 1995] to the time interval domain. This method applies the Apriori pruning in the candidate generation phase to prune any candidate *k-pattern* that contains an infrequent *(k-1)-patterns* (see Section 2.2.1).
2. ***TP_lists:*** Mine frequent temporal patterns by extending the vertical format [Zaki, 2000, Zaki, 2001] to the time interval domain as described in [Batal et al., 2011]. This method applies the Apriori pruning in candidate generation and uses the *id-list* indexing to speed up the counting.
3. ***RTP_no-lists:*** Mine frequent RTPs backward in time as described in Section 5.4, but

without applying the technique described in Section 5.4.3 to speed up the counting. That is, this method scans the entire dataset for each generated candidate in order to compute its *RTP-sup*.

4. ***RTP_lists:*** Our proposed method for mining frequent RTPs.
5. ***MPRTP:*** Our proposed method for mining MPRTPs. This method applies both the lossless pruning and the lossy pruning to restrict the search space of temporal patterns (see Section 5.5).

To make the comparison fair, *all methods apply the techniques we propose in Section 5.4.2 to avoid generating incoherent candidates*. Note that if we do not remove incoherent candidates, the execution time for all methods greatly increases.

The experiments are conducted on a Dell Precision T1600 machine with an Intel Xeon 3GHz CPU and 16GB of RAM. All methods are implemented in MATLAB.

5.7.4.2 Results on Synthetic Data Similar to the previous settings for the synthetic data (Section 5.7.2.2), we set the local minimum supports to 10% and the maximum gap parameter to 1 time unit.

Figure 39 shows the execution time (on logarithmic scale) of the compared methods. We can see that *RTP-lists* and *MPRTP* are the most efficient methods.

5.7.4.3 Results on HIT Data Similar to the previous settings for the HIT data (Section 5.7.2.3), we set the local minimum supports to 10% and the maximum gap parameter to 2 days.

Figure 40 shows the execution time (on logarithmic scale) of the compared methods. Again, we see that *RTP-lists* and *MPRTP* outperform the other temporal pattern mining methods.

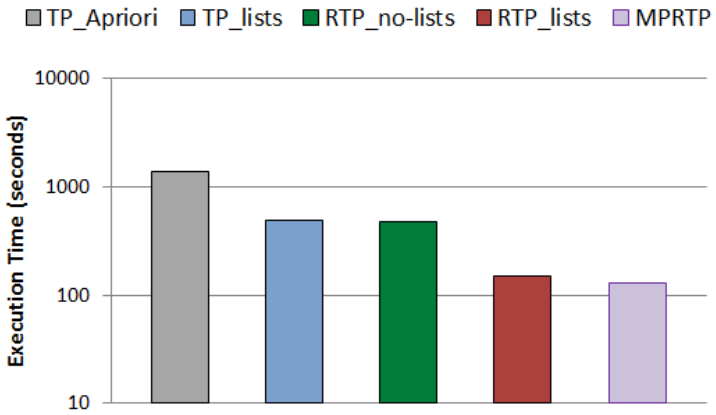


Figure 39: **Synthetic dataset**: The mining time (in seconds) of the compared temporal pattern mining methods (Section 5.7.4.1). The local minimum support is 10%.

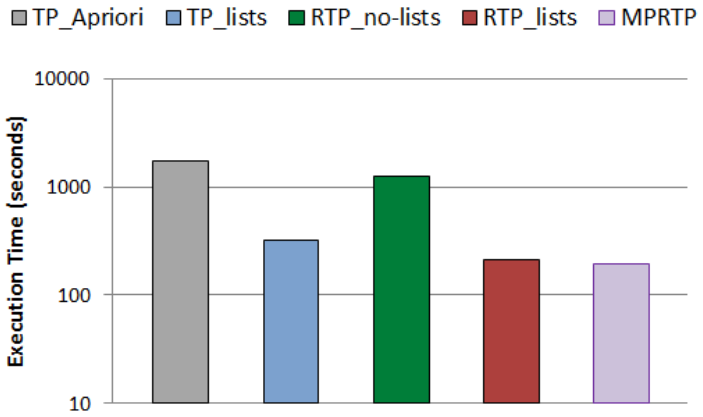


Figure 40: **HIT dataset**: The mining time (in seconds) of the compared temporal pattern mining methods (Section 5.7.4.1). The local minimum support is 10%.

5.7.4.4 Results on Diabetes Data Similar to the previous settings for the diabetes data (Section 5.7.2.4), we set the local minimum supports to 15% and the maximum gap parameter to 6 months (unless stated otherwise).

Figure 41 shows the execution time (on logarithmic scale) of the compared methods on all major diagnosis datasets. We can see that *RTP_lists* and *MPRTP* are much more efficient than the other temporal pattern mining methods. For example, on the *INFLM* dataset, *RTP_lists* is around 5 times faster than *TP_lists*, 10 times faster than *RTP_no-lists* and 30 times faster than *TP_Apriori*. Furthermore, *MPRTP* is more efficient than *RTP_lists* for all datasets.

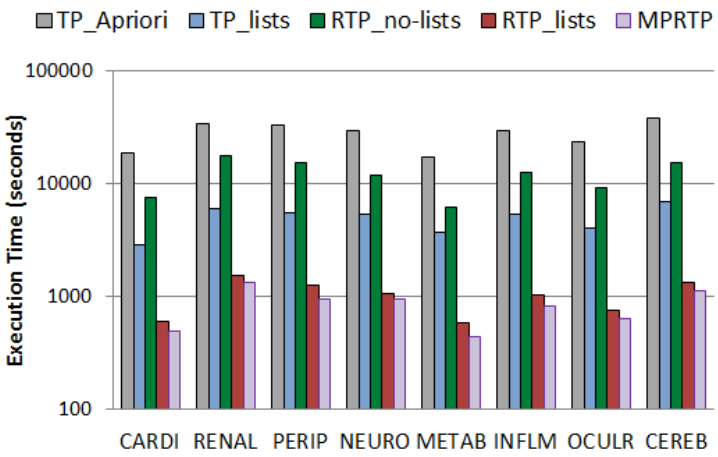


Figure 41: **Diabetes dataset**: The mining time (in seconds) of the compared temporal pattern mining methods (Section 5.7.4.1) for the eight major diabetes diseases. The local minimum support is 15%.

Figure 42 compares the execution time of the different methods on the *CARDI* dataset for different minimum support thresholds.

Finally, let us examine the effect of the maximum gap parameter (g) on the efficiency of recent temporal pattern mining methods (*RTP_no-lists*, *RTP_lists* and *MPRTP*). Figure 43 shows the execution time on the *CARDI* dataset for different values of g (the execution time of *TP_Apriori* and *TP_lists* does not depend of g).

Clearly, the execution time of *RTP_no-lists*, *RTP_lists* and *MPRTP* increase with g because the search space becomes larger (more temporal patterns become RTPs). We can see that when g is more than 18 months, *RTP_no-lists* becomes slower than *TP_Apriori*. The reason is that for large values of g, applying the Apriori pruning in candidate generation

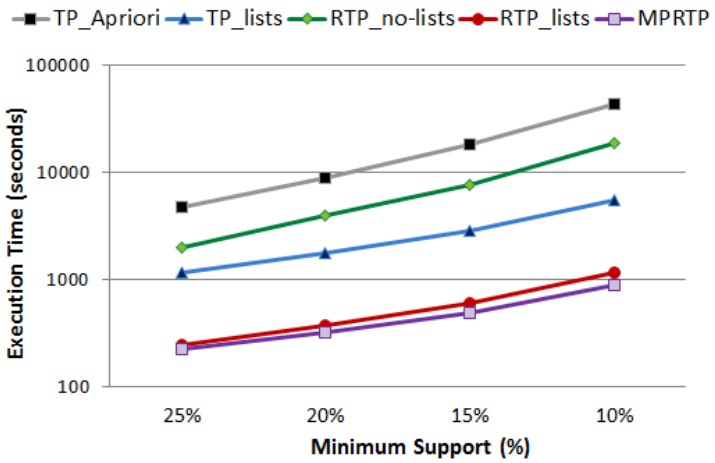


Figure 42: **Diabetes dataset (CARDI)**: The mining time (in seconds) of the compared temporal pattern mining methods (Section 5.7.4.1) on the CARDI diabetes dataset for different values of the minimum support.

becomes more efficient (generates less candidates) than the backward extension of temporal patterns (see Example 15). On the other hand, the execution time of *RTP_lists* and *MPRTP* increase much slower with g and they maintain their efficiency advantage over *TP_Apriori* and *TP_lists* for larger values of g.

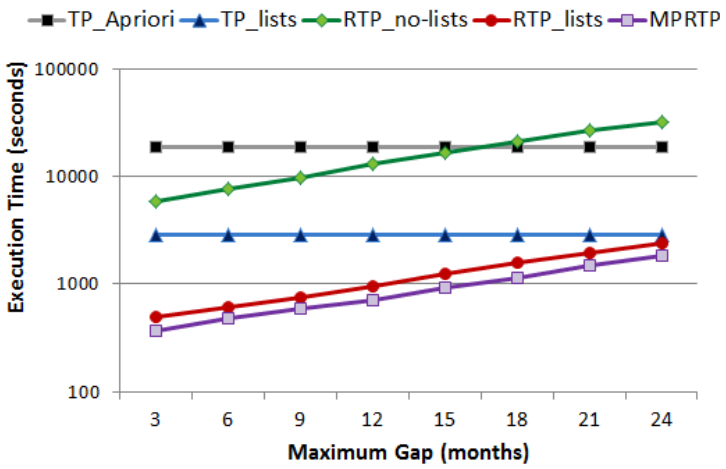


Figure 43: **Diabetes dataset (CARDI)**: The mining time (in seconds) of the compared temporal pattern mining methods (Section 5.7.4.1) on the CARDI diabetes dataset for different values of the maximum gap parameter.

5.8 SUMMARY

In this chapter, we studied the problem of mining predictive temporal patterns in complex multivariate time series data, such as electronic health records. We used temporal abstraction and temporal logic for defining and representing the temporal patterns.

It is well known that mining the entire set of frequent temporal patterns (whether sequential patterns or time-interval patterns) from large-scale data is computationally very expensive. To alleviate this problem, previous research [Srikant and Agrawal, 1996, Pei et al., 2007] introduced several temporal constraints to scale up the mining, such as restricting the overall pattern duration or restricting the permitted gap between consecutive states in a pattern. This chapter proposed a new class of temporal constraints for finding recent temporal patterns (RTP), which we argued is appropriate for event detection problems. We presented an efficient algorithm that mines time-interval patterns backward in time, starting from patterns related to most recent observations. We also presented the

minimal predictive recent temporal patterns (MPRTP) framework for selecting predictive and non-spurious RTPs.

We tested and demonstrated the usefulness of our framework on two real-world clinical tasks. The first is to predict patients who are at risk of developing heparin induced thrombocytopenia, a life threatening condition that may develop in patients treated with heparin. The second is predict and diagnose various disorders for diabetic patients, such as cardiological, renal or neurological disorders. Our experimental evaluation demonstrated the following benefits of our approach:

1. RTP mining and MPRTP mining are able to learn accurate event detection classifiers for real-world clinical tasks, which is a key step for developing intelligent clinical monitoring systems.
2. The MPRTP framework is effective for selecting predictive and non-spurious RTPs, which makes it useful for knowledge discovery.
3. Mining RTPs or MPRTPs is more scalable than the existing temporal pattern mining methods.

6.0 DISCUSSION

In this dissertation, we studied pattern mining in the supervised setting, where the objective is to find patterns (defining subpopulations of the data) that are important for predicting the class labels. We have presented several methods for mining predictive patterns for both atemporal and temporal data. The main contributions of this dissertation are summarized below.

- We presented the minimal predictive patterns (MPP) framework for supervised pattern mining in static (atemporal) data. This framework applies a novel Bayesian score to evaluate the predictiveness of patterns. It also considers the structure of patterns to assure that every pattern is not only predictive compared to the entire data, but also compared to the data matching any of its subpatterns. We showed that the MPP framework is able to explain and summarize the data using fewer patterns that the existing methods. We also showed that using MPPs as features can greatly improve the classification performance.
- We presented an efficient algorithm for mining MPPs, which integrates pattern evaluation with frequent pattern mining and applies supervised pruning strategies to speed up the mining. We showed that our algorithm is more efficient than standard frequent pattern mining algorithms.
- We also studied the problem of supervised pattern mining in multivariate temporal data. We presented a novel method for mining recent temporal patterns (RTP), which we argued is appropriate for event detection problems. We showed that the RTP framework is able to learn accurate event detection models for real-world clinical tasks. In addition, we showed that it is much more efficient and scalable than existing temporal pattern

mining methods.

- We extended the MPP framework to the temporal domain and presented the minimal predictive recent temporal patterns (MPRTP). We showed that MPRTP is effective for selecting predictive and non-spurious RTPs.

There are however some limitations of pattern mining techniques, which our proposed methods inherit:

- Pattern mining suffers when applied on high-dimensional data. The reason is that when the dimensionality of the data is large, the space of patterns becomes very large, which in turn makes the mining computationally very expensive and increases the risk of false discoveries.
- Pattern mining requires a prior discretization of the data in order to convert numeric values to a finite number of categories. This may result in loosing some predictive information in the numeric attributes. Besides, pattern mining treats these discretized categories as being independent and disregards their ordinal relations.

We now outline some related open questions and research opportunities.

- **Mining Association Rules:** This is an *unsupervised* pattern mining task which aims to extract interesting correlations, associations and casual relations between items in the data[1]. Association rules are usually obtained by first applying a frequent pattern mining method and then generating rules that have coverage and confidence higher than user-specified thresholds [Han et al., 2006]. However, using a similar argument to the one in Section 3.4, we can see that this approach usually leads to many spurious association rules. For example, if rule $chips \Rightarrow salsa$ has a high confidence, many of its spurious rules, such as $chips \wedge banana \Rightarrow salsa$, are expected to have high confidences as well.

  The MPP framework we proposed for supervised pattern mining can also be used to filter out spurious association rules. That is, we can apply it as a postprocessing step to

[1] In contrast to our work, where we restrict the consequent of rules to be a class label (supervised), the consequent of rules for association rule mining can be any item (unsupervised).

assure that every association rule in the result offers a significant predictive advantage over all of its subrules.

- **Comparing and Contrasting Datasets:** Identifying and explaining the similarities and differences between two datasets can be very valuable. For example, suppose we have data about patients in two different intensive care units (ICUs), or within the same ICU during two different periods. If the two ICUs experience different outcomes (e.g., different mortality rates), we may wish to understand and gain insights on the reasons they differ[2]. An important research problem is to extend our method to search for patterns that most contribute to the differences between datasets and provide explanations on how they account for the differences.
- **Detecting Patterns in Spatio-Temporal Data:** The aim of this task is to find patterns that describe the temporal changes in the relations between spatially related objects. For example, assume we have a temporal sequence of medical images and an object detection algorithm. Assume we detected two neighboring objects A and B and defined their relations using the intensity gradient. It would be interesting to study patterns that describe how this relation changes over time. An example of such patterns is Intensity_gradient(A,B)=low proceeds Intensity_gradient(A,B)=high.
- **Mining Pattern Sets:** Traditional pattern mining methods are based on the idea of evaluating the quality of *individual patterns* and choosing the top quality ones. In this thesis, we proposed a method that considers the relations between patterns (the partial order defined on the lattice of patterns) when evaluating their quality. An alternative (and more general) approach is to cast pattern mining as an optimization problem. This can be done by specifying a function that evaluates the quality of an entire *set of patterns* and finding a set that optimizes (or satisfies constraints on) that function. An example of such task is to find the *smallest* set of patterns that *collectively* cover at least 90% of the data and predict the class label with accuracy at least 80%. This general formulation appears to be hard to solve. An interesting research direction is to investigate specific forms of quality functions that make the problem computationally more tractable.

[2]For example, the higher mortality in hospital A compared to hospital B may be simply because patients in A were in worse conditions than patients in B, not because of worse patient management.

APPENDIX

MATHEMATICAL DERIVATION AND COMPUTATIONAL COMPLEXITY OF THE BAYESIAN SCORE

This appendix explains in details the mathematical derivations and the computational complexity of the Bayesian score described in Section 3.5.1.2. Section A.2 derives the closed form solution for the marginal likelihood of model M_h: $P(G|M_h)$. Section A.3 shows the four equivalent formulas for solving $P(G|M_h)$. Section A.4 illustrates how to obtain the marginal likelihood of model M_l from the solution to the marginal likelihood of model M_h. Finally, Section A.5 analyzes the overall computational complexity for computing the Bayesian score.

A.1 DEFINITION AND NOTATIONS

We want to evaluate rule $P \Rightarrow y$ with respect to a group of instances G where $G_P \subseteq G$. Let Y denote the class variable (the outcome). Let θ denote the probability of class $Y = y$ in G, let θ_1 denote the probability of $= y$ in G_P and let θ_2 denote the probability of $= y$ in the instances of G not covered by P ($G \setminus G_P$).

We define the following three models:

1. $\boldsymbol{M_e}$ is the model that conjectures that θ_1 is the **same** as θ_2.
2. $\boldsymbol{M_h}$ is the model that conjectures that θ_1 is **higher** than θ_2.

3. $\boldsymbol{M_l}$ is the model that conjectures that θ_1 is **lower** than θ_2.

Let α and β be the beta parameters for the prior distribution on θ. Let α_1 and β_1 be the beta parameters for the prior distribution on θ_1. Let α_2 and β_2 be the beta parameters for the prior distribution on θ_2. Let N_{*1} and N_{*2} be the number of instances in G with $Y = y$ and with $Y \neq y$, respectively. Let N_{11} and N_{12} be the number of instances in G_P with $Y = y$ and with $Y \neq y$, respectively. Let N_{21} and N_{22} be the number of instances in $G \setminus G_P$ with $Y = y$ and with $Y \neq y$, respectively.

We define the Bayesian score of rule $P \Rightarrow y$ with respect to group G as follows:

$$BS(P \Rightarrow y, G) = Pr(M_h|G) = \frac{Pr(G|M_h) \cdot Pr(M_h)}{Pr(G|M_e) \cdot Pr(M_e) + Pr(G|M_h) \cdot Pr(M_h) + Pr(G|M_l) \cdot Pr(M_l)} \quad (.1)$$

Evaluating Equation .1 requires evaluating the marginal likelihood for models M_e, M_h and M_l. Evaluating the marginal likelihood of M_e is easy and is given by the following well known closed-form solution [Heckerman et al., 1995]:

$$Pr(G|M_e) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha+N_{*1}+\beta+N_{*2})} \cdot \frac{\Gamma(\alpha+N_{*1})}{\Gamma(\alpha)} \cdot \frac{\Gamma(\beta+N_{*2})}{\Gamma(\beta)} \quad (.2)$$

where Γ is the gamma function.

In the rest of this appendix, we describe how to obtain closed-form solutions for the marginal likelihood of M_h and M_l and then analyse the overall computational complexity of the Bayesian score (evaluating Equation .1).

A.2 DERIVATION OF THE CLOSED-FORM SOLUTION FOR MODEL M_H

The marginal likelihood of model M_h ($Pr(G|M_h)$) is defined as follows:

$$= \frac{1}{k} \int_{\theta_1=0}^{1} \int_{\theta_2=0}^{\theta_1} {\theta_1}^{N_{11}} \cdot (1-\theta_1)^{N_{12}} \cdot {\theta_2}^{N_{21}} \cdot (1-\theta_2)^{N_{22}} \cdot beta(\theta_1;\alpha_1,\beta_1) \cdot beta(\theta_2;\alpha_2,\beta_2) d\theta_2 d\theta_1$$

$$= \frac{1}{k} \underbrace{\int_{\theta_1=0}^{1} {\theta_1}^{N_{11}} \cdot (1-\theta_1)^{N_{12}} \cdot beta(\theta_1;\alpha_1,\beta_1)}_{f_1} \underbrace{\int_{\theta_2=0}^{\theta_1} {\theta_2}^{N_{21}} \cdot (1-\theta_2)^{N_{22}} \cdot beta(\theta_2;\alpha_2,\beta_2) d\theta_2}_{f_2} d\theta_1 \tag{.3}$$

We first show how to solve the integral over θ_2 in closed form, which is denoted by f_2 in Equation .3. We then expand the function denoted by f_1, multiply it by the solution to f_2, and solve the integral over θ_1 in closed form to complete the integration.

We use the regularized incomplete beta function [Abramowitz and Stegun, 1964] to solve the integral given by f_2. Using the notation in the expression denoted by f_2, the incomplete beta function is as follows:

$$\int_{\theta_2=0}^{\theta_1} {\theta_2}^{a-1} \cdot (1-\theta_2)^{b-1} d\theta_2 = \frac{\Gamma(a) \cdot \Gamma(b)}{\Gamma(a+b)} \cdot \sum_{j=a}^{a+b-1} \frac{\Gamma(a+b)}{\Gamma(j+1) \cdot \Gamma(a+b-j)} \cdot \theta_1^j \cdot (1-\theta_1)^{a+b-1-j} \tag{.4}$$

where a and b should be natural numbers.

Note that when $\theta_1 = 1$ in Equation .4, the solution to the integral in that equation is simply the following:

$$\int_{\theta_2=0}^{1} {\theta_2}^{a-1} \cdot (1-\theta_2)^{b-1} d\theta_2 = \frac{\Gamma(a) \cdot \Gamma(b)}{\Gamma(a+b)} \tag{.5}$$

We now solve the integral given by f_2 in Equation .3 as follows:

$$f_2 = \int_{\theta_2=0}^{\theta_1} {\theta_2}^{N_{21}} \cdot (1-\theta_2)^{N_{22}} \cdot beta(\theta_2;\alpha_2,\beta_2) d\theta_2$$

$$= \int_{\theta_2=0}^{\theta_1} {\theta_2}^{N_{21}} \cdot (1-\theta_2)^{N_{22}} \cdot \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2) \cdot \Gamma(\beta_2)} \cdot \theta_2^{\alpha_2-1} \cdot (1-\theta_2)^{\beta_2-1} d\theta_2$$

$$= \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2) \cdot \Gamma(\beta_2)} \int_{\theta_2=0}^{\theta_1} {\theta_2}^{N_{21}+\alpha_2-1} \cdot (1-\theta_2)^{N_{22}+\beta_2-1} d\theta_2$$

$$= \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2) \cdot \Gamma(\beta_2)} \int_{\theta_2=0}^{\theta_1} {\theta_2}^{a-1} \cdot (1-\theta_2)^{b-1} d\theta_2$$

where $a = N_{21} + \alpha_2$ and $b = N_{22} + \beta_2$.

Using Equation .4, we get the following:

$$f_2 = \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2)\cdot\Gamma(\beta_2)} \cdot \frac{\Gamma(a)\cdot\Gamma(b)}{\Gamma(a+b)} \cdot \sum_{j=a}^{a+b-1} \frac{\Gamma(a+b)}{\Gamma(j+1)\cdot\Gamma(a+b-j)} \cdot \theta_1^j \cdot (1-\theta_1)^{a+b-1-j} \tag{.6}$$

We now turn to f_1, which can be expanded as follows:

$$\begin{aligned} f_1 &= \int_{\theta_1=0}^{1} {\theta_1}^{N_{11}} \cdot (1-\theta_1)^{N_{12}} \cdot beta(\theta_1;\alpha_1,\beta_1) \\ &= \int_{\theta_1=0}^{1} {\theta_1}^{N_{11}} \cdot (1-\theta_1)^{N_{12}} \cdot \frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\cdot\Gamma(\beta_1)} \cdot \theta_1^{\alpha_1-1} \cdot (1-\theta_1)^{\beta_1-1} \qquad (.7) \\ &= \frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\cdot\Gamma(\beta_1)} \int_{\theta_1=0}^{1} {\theta_1}^{c-1} \cdot (1-\theta_1)^{d-1} \end{aligned}$$

where $c = N_{11} + \alpha_1$ and $d = N_{12} + \beta_1$.

Now we combine Equations .6 and .7 to solve Equation .3:

$$\begin{aligned} Pr(G|M_h) &= \frac{1}{k} \cdot f_1 \cdot f_2 \, d\theta_1 \\ &= \frac{1}{k} \cdot \frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\cdot\Gamma(\beta_1)} \cdot \int_{\theta_1=0}^{1} {\theta_1}^{c-1} \cdot (1-\theta_1)^{d-1} \cdot \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2)\cdot\Gamma(\beta_2)} \cdot \frac{\Gamma(a)\cdot\Gamma(b)}{\Gamma(a+b)} \\ &\quad \cdot \sum_{j=a}^{a+b-1} \frac{\Gamma(a+b)}{\Gamma(j+1)\cdot\Gamma(a+b-j)} \cdot \theta_1^j \cdot (1-\theta_1)^{a+b-1-j} d\theta_1 \\ &= \frac{1}{k} \cdot \frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\cdot\Gamma(\beta_1)} \cdot \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2)\cdot\Gamma(\beta_2)} \cdot \frac{\Gamma(a)\cdot\Gamma(b)}{\Gamma(a+b)} \cdot \int_{\theta_1=0}^{1} {\theta_1}^{c-1} \cdot (1-\theta_1)^{d-1} \\ &\quad \cdot \sum_{j=a}^{a+b-1} \frac{\Gamma(a+b)}{\Gamma(j+1)\cdot\Gamma(a+b-j)} \cdot \theta_1^j \cdot (1-\theta_1)^{a+b-1-j} d\theta_1 \\ &= \frac{1}{k} \cdot \frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\cdot\Gamma(\beta_1)} \cdot \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2)\cdot\Gamma(\beta_2)} \cdot \frac{\Gamma(a)\cdot\Gamma(b)}{\Gamma(a+b)} \\ &\quad \cdot \sum_{j=a}^{a+b-1} \frac{\Gamma(a+b)}{\Gamma(j+1)\cdot\Gamma(a+b-j)} \cdot \int_{\theta_1=0}^{1} {\theta_1}^{(c+j)-1} \cdot (1-\theta_1)^{(a+b+d-1-j)-1} d\theta_1 \end{aligned}$$

Which by Equation .5 is

$$Pr(G|M_h) = \frac{1}{k} \cdot \frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\cdot\Gamma(\beta_1)} \cdot \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2)\cdot\Gamma(\beta_2)} \cdot \frac{\Gamma(a)\cdot\Gamma(b)}{\Gamma(a+b)}$$

$$\cdot \sum_{j=a}^{a+b-1} \frac{\Gamma(a+b)}{\Gamma(j+1)\cdot\Gamma(a+b-j)} \cdot \frac{\Gamma(c+j)\cdot\Gamma(a+b+d-1-j)}{\Gamma(a+b+c+d-1)}$$

$$= \frac{1}{k} \cdot \frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\cdot\Gamma(\beta_1)} \cdot \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2)\cdot\Gamma(\beta_2)} \cdot \sum_{j=a}^{a+b-1} \frac{\Gamma(a)\cdot\Gamma(b)}{\Gamma(j+1)\cdot\Gamma(a+b-j)} \cdot \frac{\Gamma(c+j)\cdot\Gamma(a+b+d-1-j)}{\Gamma(a+b+c+d-1)} \tag{.8}$$

where $a = N_{21} + \alpha_2$, $b = N_{22} + \beta_2$, $c = N_{11} + \alpha_1$ and $d = N_{12} + \beta_1$.

We can solve for k (the normalization constant for the parameter prior) by solving Equation .3 (without the k term) with $N_{11} = N_{12} = N_{21} = N_{22} = 0$. Doing so is equivalent to applying Equation .8 (without the k term) with $a = \alpha_2$, $b = \beta_2$, $c = \alpha_1$ and $d = \beta_1$. Note that $k = \frac{1}{2}$ if we use uniform priors on both parameters by setting $\alpha_1 = \beta_1 = \alpha_2 = \beta_2 = 1$.

A.3 FOUR EQUIVALENT SOLUTIONS FOR MODEL M_H

In the previous section, we showed the full derivation of the closed-form solution to the marginal likelihood of model M_h. It turned out that there are four equivalent solutions to Equation .3. These solutions are derived by redefining which class map to the values 1 and 2 and by redefining which regions map to θ_1 and θ_2.

Let us use the notations introduced in the previous section: $a = N_{21} + \alpha_2$, $b = N_{22} + \beta_2$, $c = N_{11} + \alpha_1$ and $d = N_{12} + \beta_1$. Also, let us define C as follows:

$$C = \frac{1}{k} \cdot \frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\cdot\Gamma(\beta_1)} \cdot \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2)\cdot\Gamma(\beta_2)} \tag{.9}$$

The marginal likelihood of M_h (Equation .3) can be obtained by solving any of the following four equations:

$$C \cdot \sum_{j=a}^{a+b-1} \frac{\Gamma(a)\cdot\Gamma(b)}{\Gamma(j+1)\cdot\Gamma(a+b-j)} \cdot \frac{\Gamma(c+j)\cdot\Gamma(a+b+d-j-1)}{\Gamma(a+b+c+d-1)} \tag{.10}$$

Which is the solution we derived in the previous section.

$$C \cdot \sum_{j=d}^{d+c-1} \frac{\Gamma(c)\cdot\Gamma(d)}{\Gamma(j+1)\cdot\Gamma(c+d-j)} \cdot \frac{\Gamma(b+j)\cdot\Gamma(c+d+a-j-1)}{\Gamma(a+b+c+d-1)} \tag{.11}$$

$$C \cdot \left(\frac{\Gamma(a)\cdot\Gamma(b)}{\Gamma(a+b)} \cdot \frac{\Gamma(c)\cdot\Gamma(d)}{\Gamma(c+d)} - \sum_{j=b}^{a+b-1} \frac{\Gamma(a)\cdot\Gamma(b)}{\Gamma(j+1)\cdot\Gamma(a+b-j)} \cdot \frac{\Gamma(d+j)\cdot\Gamma(a+b+c-j-1)}{\Gamma(a+b+c+d-1)} \right) \tag{.12}$$

$$C \cdot \left(\frac{\Gamma(a)\cdot\Gamma(b)}{\Gamma(a+b)} \cdot \frac{\Gamma(c)\cdot\Gamma(d)}{\Gamma(c+d)} - \sum_{j=c}^{c+d-1} \frac{\Gamma(c)\cdot\Gamma(d)}{\Gamma(j+1)\cdot\Gamma(c+d-j)} \cdot \frac{\Gamma(a+j)\cdot\Gamma(c+d+b-j-1)}{\Gamma(a+b+c+d-1)} \right) \tag{.13}$$

A.4 DERIVATION OF THE CLOSED-FORM SOLUTION FOR MODEL M_L

The marginal likelihood of model M_l ($Pr(G|M_l)$) is defined as follows:

$$= \frac{1}{k} \underbrace{\int_{\theta_2=0}^{1} {\theta_2}^{N_{21}} \cdot (1-\theta_2)^{N_{22}} \cdot beta(\theta_2;\alpha_2,\beta_2)}_{f_1} \underbrace{\int_{\theta_1=0}^{\theta_2} {\theta_1}^{N_{11}} \cdot (1-\theta_1)^{N_{11}} \cdot beta(\theta_1;\alpha_1,\beta_1) d\theta_1}_{f_2} d\theta_2 \tag{.14}$$

By solving the integral given by f_2, we get:

$$f_2 = \frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\cdot\Gamma(\beta_1)} \int_{\theta_1=0}^{\theta_2} {\theta_1}^{c-1} \cdot (1-\theta_1)^{d-1} d\theta_2$$

$$= \frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\cdot\Gamma(\beta_1)} \cdot \frac{\Gamma(c)\cdot\Gamma(d)}{\Gamma(c+d)} \cdot \sum_{j=c}^{c+d-1} \frac{\Gamma(c+d)}{\Gamma(j+1)\cdot\Gamma(c+d-j)} \cdot \theta_2^j \cdot (1-\theta_2)^{c+d-1-j}$$

where, as before, $c = N_{11} + \alpha_1$ and $d = N_{12} + \beta_1$.

By solving f_1, we get:

$$f_1 = \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2)\cdot\Gamma(\beta_2)} \int_{\theta_2=0}^{1} {\theta_2}^{a-1} \cdot (1-\theta_2)^{b-1}$$

Now we can solve Equation .14:

$$Pr(G|M_l) = C \cdot \sum_{j=c}^{c+d-1} \frac{\Gamma(c)\cdot\Gamma(d)}{\Gamma(j+1)\cdot\Gamma(c+d-j)} \cdot \frac{\Gamma(a+j)\cdot\Gamma(c+d+b-1-j)}{\Gamma(a+b+c+d-1)} \quad (.15)$$

Where C is the constant we defined by Equation .9 in the previous section.

Notice that Equation .15 (the solution to $Pr(G|M_l)$) can be obtained from Equation .13 (one of the four solutions to $Pr(G|M_h)$) as follows:

$$Pr(G|M_l) = C \cdot \frac{\Gamma(a)\cdot\Gamma(b)\Gamma(c)\cdot\Gamma(d)}{\Gamma(a+b)\cdot -\Gamma(c+d)} - Pr(G|M_h) \quad (.16)$$

It turned out that no matter which formula we used to solve $Pr(G|M_h)$, we can use Equation .16 to obtain $Pr(G|M_l)$.

A.5 COMPUTATIONAL COMPLEXITY

Since we require that N_{11}, N_{12}, N_{21}, N_{22}, α_1, β_1, α_2 and β_2 be natural numbers, the gamma function simply becomes a factorial function: $\Gamma(x) = (x-1)!$. Since such numbers can become very large, it is convenient to use the logarithm of the gamma function and express Equations .2, .10, .11, .12, .13 and .16 in logarithmic form in order to preserve numerical precision. The logarithm of the integer gamma function can be pre-computed and efficiently stored in an array as follows:

$lnGamma[1] = 0$

For $i = 2$ to n

$lnGamma[i] = lnGamma[i-1] + ln(i-1)$

We then can use $lnGamma$ in solving the above equations. However, Equations .10, .11, .12 and .13 include a sum, which makes the use of the logarithmic form more involved. To deal with this issue, we can define function $lnAdd$, which takes two arguments x and y that are in logarithmic form and returns $ln(e^x + e^y)$. It does so in a way that preserves a good deal of numerical precision that could be lost if $ln(e^x + e^y)$ were calculated in a direct

manner. This is done by using the following formula:

$$lnAdd(x,y) = x + ln(1 + e^{(y-x)})$$

Now that we introduced functions $lnGamma$ and $lnAdd$, it is straightforward to evaluate Equations .2, .10, .11, .12, .13 and .16 in logarithmic form.

Let us now analyze the overall computational complexity for computing the Bayesian score for a specific rule (solving Equation .1). Doing so requires computing $Pr(M_e|G)$, $Pr(M_h|G)$ and $Pr(M_l|G)$. $Pr(M_e|G)$ can be computed in $O(1)$ using Equation .2. $Pr(M_h|G)$ can be computed by applying Equation .10, Equation .11, Equation .12 or Equation .13. The computational complexity of these equations are $O(N_{22}+\beta_2)$, $O(N_{11}+\alpha_1)$, $O(N_{21}+\alpha_2)$ and $O(N_{12}+\beta_1)$, respectively. Therefore, $Pr(M_h|G)$ can be computed in $O(min(N_{11}+\alpha_1, N_{12}+\beta_1, N_{21}+\alpha_2, N_{22}+\beta_2)$. $Pr(M_l|G)$ can be computed from $Pr(M_h|G)$ in $O(1)$ using Equation .16. By assuming that α_1, β_1, α_2, β_2 are bounded from above, the overall complexity for computing the Bayesian score is $\boldsymbol{O(min(N_{11}, N_{12}, N_{21}, N_{22})}$.

BIBLIOGRAPHY

[Abramowitz and Stegun, 1964] Abramowitz, M. and Stegun, I. A. (1964). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*.

[Agrawal et al., 1998] Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998). Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *Proceedings of the international conference on Management Of Data (SIGMOD)*.

[Agrawal et al., 1993] Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the international conference on Management Of Data (SIGMOD)*.

[Agrawal and Shafer, 1996] Agrawal, R. and Shafer, J. C. (1996). Parallel Mining of Association Rules. *IEEE Transaction on Knowledge and Data Engineering*, 8:962–969.

[Agrawal and Srikant, 1994] Agrawal, R. and Srikant, R. (1994). Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the international conference on Very Large Data Bases (VLDB)*.

[Agrawal and Srikant, 1995] Agrawal, R. and Srikant, R. (1995). Mining Sequential Patterns. In *Proceedings of the International Conference on Data Engineering (ICDE)*.

[Allen, 1984] Allen, F. (1984). Towards a General Theory of Action and Time. *Artificial Intelligence, 23:123-154*.

[Asuncion and Newman, 2007] Asuncion, A. and Newman, D. (2007). UCI machine learning repository.

[Bailey et al., 2002] Bailey, J., Manoukian, T., and Ramamohanarao, K. (2002). Fast Algorithms for Mining Emerging Patterns. In *Proceedings of the European conference on Principles of Data Mining and Knowledge Discovery (PKDD)*.

[Batal et al., 2012a] Batal, I., Cooper, G., and Hauskrecht, M. (2012a). A Bayesian Scoring Technique for Mining Predictive and Non-Spurious Rules. In *Proceedings of the European conference on Principles of Data Mining and Knowledge Discovery (PKDD)*.

[Batal et al., 2012b] Batal, I., Fradkin, D., Harrison, J., Moerchen, F., and Hauskrecht, M. (2012b). Mining Recent Temporal Patterns for Event Detection in Multivariate Time

Series Data. In *Proceedings of the international conference on Knowledge Discovery and Data mining (SIGKDD)*.

[Batal and Hauskrecht, 2009] Batal, I. and Hauskrecht, M. (2009). A Supervised Time Series Feature Extraction Technique Using DCT and DWT. In *International Conference on Machine Learning and Applications (ICMLA)*.

[Batal and Hauskrecht, 2010a] Batal, I. and Hauskrecht, M. (2010a). A Concise Representation of Association Rules using Minimal Predictive Rules. In *Proceedings of the European conference on Principles of Data Mining and Knowledge Discovery (PKDD)*.

[Batal and Hauskrecht, 2010b] Batal, I. and Hauskrecht, M. (2010b). Constructing Classification Features using Minimal Predictive Patterns. In *Proceedings of the international conference on Information and knowledge management (CIKM)*.

[Batal et al., 2009] Batal, I., Sacchi, L., Bellazzi, R., and Hauskrecht, M. (2009). Multivariate Time Series Classification with Temporal Abstractions. In *Proceedings of the Florida Artificial Intelligence Research Society (FLAIRS)*.

[Batal et al., 2011] Batal, I., Valizadegan, H., Cooper, G. F., and Hauskrecht, M. (2011). A Pattern Mining Approach for Classifying Multivariate Temporal Data. In *Proceedings of the IEEE international conference on bioinformatics and biomedicine (BIBM)*.

[Batal et al., 2012c] Batal, I., Valizadegan, H., Cooper, G. F., and Hauskrecht, M. (2012c). A Temporal Pattern Mining Approach for Classifying Electronic Health Record Data. *ACM Transaction on Intelligent Systems and Technology (ACM TIST), Special Issue on Health Informatics*.

[Bay and Pazzani, 2001] Bay, S. D. and Pazzani, M. J. (2001). Detecting Group Differences: Mining Contrast Sets. *Data Mining and Knowledge Discovery*, 5:213–246.

[Bayardo, 1998] Bayardo, R. J. (1998). Efficiently Mining Long Patterns from Databases. In *Proceedings of the international conference on Management Of Data (SIGMOD)*.

[Bayardo, 1999] Bayardo, R. J. (1999). Constraint-Based Rule Mining in Large, Dense Databases. In *Proceedings of the International Conference on Data Engineering (ICDE)*.

[Beil et al., 2002] Beil, F., Ester, M., and Xu, X. (2002). Frequent Term-based Text Clustering. In *Proceedings of the international conference on Knowledge discovery and data mining (SIGKDD)*.

[Benjamini and Hochberg, 1995] Benjamini, Y. and Hochberg, Y. (1995). Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society*, 57(1):289–300.

[Blasiak and Rangwala, 2011] Blasiak, S. and Rangwala, H. (2011). A Hidden Markov Model Variant for Sequence Classification. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*.

[Breiman et al., 1984] Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company.

[Brin et al., 1997a] Brin, S., Motwani, R., and Silverstein, C. (1997a). Beyond Market Baskets: Generalizing Association Rules to Correlations. In *Proceedings of the international conference on Management Of Data (SIGMOD)*.

[Brin et al., 1997b] Brin, S., Motwani, R., Ullman, J. D., and Tsur, S. (1997b). Dynamic Itemset Counting and Implication Rules for Market Basket Data. In *Proceedings of the international conference on Management Of Data (SIGMOD)*.

[Calders and Goethals, 2002] Calders, T. and Goethals, B. (2002). Mining All Non-derivable Frequent Itemsets. In *Proceedings of the European conference on Principles of Data Mining and Knowledge Discovery (PKDD)*.

[Casas-Garriga, 2005] Casas-Garriga, G. (2005). Summarizing Sequential Data with Closed Partial Orders. In *Proceedings of the SIAM international conference on Data Mining (SDM)*.

[Chandola et al., 2006] Chandola, V., Eilertson, E., Ertoz, L., Simon, G., and Kumar, V. (2006). *Data Warehousing and Data Mining Techniques for Computer Security*, chapter Data Mining for Cyber Security. Springer.

[Cheng et al., 2007] Cheng, H., Yan, X., Han, J., and wei Hsu, C. (2007). Discriminative Frequent Pattern Analysis for Effective Classification. In *Proceedings of the International Conference on Data Engineering (ICDE)*.

[Cheng et al., 2008] Cheng, H., Yan, X., Han, J., and Yu, P. S. (2008). Direct Discriminative Pattern Mining for Effective Classification. In *Proceedings of the International Conference on Data Engineering (ICDE)*.

[Clark and Niblett, 1989] Clark, P. and Niblett, T. (1989). The CN2 Induction Algorithm. *Machine Learning*, 3:261–283.

[Cohen, 1995] Cohen, W. (1995). Fast Effective Rule Induction. In *Proceedings of International Conference on Machine Learning (ICML)*.

[Cohen and Singer, 1999] Cohen, W. and Singer, Y. (1999). A Simple, Fast, and Effective Rule Learner. In *Proceedings of the National conference on Artificial Intelligence (AAAI)*.

[Cong et al., 2005] Cong, G., Tan, K.-L., Tung, A. K. H., and Xu, X. (2005). Mining Top-K Covering Rule Groups for Gene Expression Data. In *Proceedings of the international conference on Management Of Data (SIGMOD)*.

[Das et al., 1998] Das, G., Lin, K.-I., Mannila, H., Renganathan, G., and Smyth, P. (1998). Rule Discovery from Time Series. In *Proceedings of the international conference on Knowledge Discovery and Data mining (SIGKDD)*.

[Deshpande et al., 2005] Deshpande, M., Kuramochi, M., Wale, N., and Karypis, G. (2005). Frequent Substructure-Based Approaches for Classifying Chemical Compounds. *IEEE Transactions on Knowledge and Data Engineering*, 17:1036–1050.

[Dong and Li, 1999] Dong, G. and Li, J. (1999). Efficient Mining of Emerging Patterns: Discovering Trends and Differences. In *Proceedings of the international conference on Knowledge discovery and data mining (SIGKDD)*.

[Exarchos et al., 2008] Exarchos, T. P., Tsipouras, M. G., Papaloukas, C., and Fotiadis, D. I. (2008). A Two-stage Methodology for Sequence Classification based on Sequential Pattern Mining and Optimization. *Data and Knowledge Engineering*, 66:467–487.

[Fan et al., 2008] Fan, W., Zhang, K., Cheng, H., Gao, J., Yan, X., Han, J., Yu, P., and Verscheure, O. (2008). Direct Mining of Discriminative and Essential Frequent Patterns via Model-based Search Tree. In *Proceeding of the international conference on Knowledge Discovery and Data mining (SIGKDD)*.

[Fawcett, 2006] Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters.*, 27(8):861–874.

[Fayyad and Irani, 1993] Fayyad, U. and Irani, K. (1993). Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

[Fischer et al., 2008] Fischer, J., Mäkinen, V., and Välimäki, N. (2008). Space Efficient String Mining under Frequency Constraints. In *Proceedings of the International Conference on Data Mining (ICDM)*.

[Friedman et al., 2000] Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive Logistic Regression: a Statistical View of Boosting. *Annals of Statistics*, 28.

[Geng and Hamilton, 2006] Geng, L. and Hamilton, H. J. (2006). Interestingness Measures for Data Mining: A Survey. *ACM Computing Surveys*, 38.

[Goethals et al., 2005] Goethals, B., Muhonen, J., and Toivonen, H. (2005). Mining Non-Derivable Association Rules. In *Proceedings of the SIAM international conference on Data Mining (SDM)*.

[Grosskreutz et al., 2010] Grosskreutz, H., Boley, M., and Krause-Traudes, M. (2010). Subgroup Discovery for Election Analysis: a Case Study in Descriptive Data Mining. In *Proceedings of the international conference on Discovery science*.

[Guttormsson et al., 1999] Guttormsson, S., Marks, R.J., I., El-Sharkawi, M., and Kerszenbaum, I. (1999). Elliptical Novelty Grouping for on-line short-turn Detection of Excited Running Rotors. *IEEE Transactions on Energy Conversion*.

[Han et al., 2007] Han, J., Cheng, H., Xin, D., and Yan, X. (2007). Frequent Pattern Mining: Current Status and Future Directions. *Data Mining and Knowledge Discovery*, 14(1):55–86.

[Han et al., 2006] Han, J., Kamber, M., and Pei, J. (2006). *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2 edition.

[Han et al., 2000] Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of the international conference on Management Of Data (SIGMOD)*.

[Hauskrecht et al., 2010] Hauskrecht, M., Valko, M., Batal, I., Clermont, G., Visweswaram, S., and Cooper, G. (2010). Conditional Outlier Detection for Clinical Alerting. In *Proceedings of the American Medical Informatics Association (AMIA)*.

[Heckerman et al., 1995] Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning*.

[Ho et al., 2003] Ho, T. B., Nguyen, T. D., Kawasaki, S., Le, S. Q., Nguyen, D. D., Yokoi, H., and Takabayashi, K. (2003). Mining Hepatitis Data with Temporal Abstraction. In *Proceedings of the international conference on Management Of Data (SIGMOD)*.

[Höppner, 2003] Höppner, F. (2003). *Knowledge Discovery from Sequential Data*. PhD thesis, Technical University Braunschweig, Germany.

[Ifrim and Wiuf, 2011] Ifrim, G. and Wiuf, C. (2011). Bounded Coordinate-descent for Biological Sequence Classification in High Dimensional Predictor Space. In *Proceedings of the international conference on Knowledge Discovery and Data mining (SIGKDD)*.

[Jain et al., 2004] Jain, A., Chang, E. Y., and Wang, Y.-F. (2004). Adaptive stream resource management using Kalman Filters. In *Proceedings of the international conference on Management Of Data (SIGMOD)*.

[Jaroszewicz and Scheffer, 2005] Jaroszewicz, S. and Scheffer, T. (2005). Fast discovery of unexpected patterns in data relative to a Bayesian network. In *Proceedings of the international conference on Knowledge discovery in data mining (SIGKDD)*.

[Kadous, 1999] Kadous, M. W. (1999). Learning Comprehensible Descriptions of Multivariate Time Series. In *Proceedings of the International Conference of Machine Learning (ICML)*.

[Kam and Fu, 2000] Kam, P.-s. and Fu, A. W.-C. (2000). Discovering Temporal Patterns for Interval-Based Events. In *Proceedings of the international conference on Data Warehousing and Knowledge Discovery (DaWaK)*.

[Kavsek and Lavrač, 2006] Kavsek, B. and Lavrač, N. (2006). APRIORI-SD: Adapting Association Rule Learning to Subgroup Discovery. *Applied Artificial Intelligence*, 20(7):543–583.

[Keogh et al., 2000] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2000). Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *journal of Knowledge and Information Systmes*, 3:263–286.

[Keogh et al., 1993] Keogh, E., Chu, S., Hart, D., and Pazzani, M. (1993). Segmenting Time Series: A Survey and Novel Approach. *Data mining in Time Series Databases*.

[Keogh et al., 2010] Keogh, E., Zhu, Q., Hu, B., Y., H., Xi, X., Wei, L., and Ratanamahatana, C. A. (2010). The UCR Time Series Classification/Clustering. http:www.cs.ucr.edu/~eamonn/time_series_data.

[Keogh et al., 2001] Keogh, E. J., Chakrabarti, K., Pazzani, M. J., and Mehrotra, S. (2001). Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *knowledge and information systems*, 3(3):263–286.

[Keogh and Pazzani, 1998] Keogh, E. J. and Pazzani, M. J. (1998). An Enhanced Representation of Time Series which Allows Fast and Accurate Classification, Clustering and Relevance Feedback. In *Proceedings of the international conference on Knowledge Discovery and Data mining (SIGKDD)*.

[Kuramochi and Karypis, 2001] Kuramochi, M. and Karypis, G. (2001). Frequent Subgraph Discovery. In *Proceedings of the International Conference on Data Mining (ICDM)*.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[Lavrač and Gamberger, 2005] Lavrač, N. and Gamberger, D. (2005). Relevancy in Constraint-based Subgroup Discovery. In *Constraint-Based Mining and Inductive Databases*.

[Lee et al., 2000] Lee, W., Stolfo, S. J., and Mok, K. W. (2000). Adaptive Intrusion Detection: A Data Mining Approach. *Artificial Intelligence Review*, 14(6):533–567.

[Leslie et al., 2002] Leslie, C. S., Eskin, E., Weston, J., and Noble, W. S. (2002). Mismatch String Kernels for SVM Protein Classification. In *Neural Information Processing Systems (NIPS)*.

[Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

[Li et al., 2001a] Li, J., Shen, H., and Topor, R. W. (2001a). Mining Optimal Class Association Rule Set. In *Proceedings of the Pacific-Asia conference on Knowledge Discovery and Data Mining (PAKDD)*.

[Li et al., 2009] Li, L., McCann, J., Pollard, N. S., and Faloutsos, C. (2009). DynaMMo: Mining and Summarization of Coevolving Sequences with Missing Values. In *Proceedings of the international conference on Knowledge Discovery and Data mining (SIGKDD)*.

[Li et al., 2010] Li, L., Prakash, B. A., and Faloutsos, C. (2010). Parsimonious Linear Fingerprinting for Time Series. *PVLDB*, 3:385–396.

[Li et al., 2001b] Li, W., Han, J., and Pei, J. (2001b). CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. In *Proceedings of the International Conference on Data Mining (ICDM)*.

[Lin et al., 2003] Lin, J., Keogh, E. J., Lonardi, S., and chi Chiu, B. Y. (2003). A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In *Proceedings of the SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*.

[Lin and Lee, 2005] Lin, M. and Lee, S. (2005). Fast Discovery of Sequential Patterns through Memory Indexing and Database Partitioning. *Journal Information Science and Engineering*, 21:109–128.

[Liu et al., 1998] Liu, B., Hsu, W., and Ma, Y. (1998). Integrating Classification and Association Rule Mining. *Knowledge Discovery and Data Mining*, pages 80–86.

[Mampaey et al., 2011] Mampaey, M., Tatti, N., and Vreeken, J. (2011). Tell me what I need to know: Succinctly Summarizing Data with Itemsets. In *Proceedings of the international conference on Knowledge Discovery and Data mining (SIGKDD)*.

[Manber and Myers, 1990] Manber, U. and Myers, G. (1990). Suffix Arrays: A New Method for On-line String Searches. In *Proceedings of the first annual SIAM symposium on Discrete algorithms*.

[Mannila et al., 1997] Mannila, H., Toivonen, H., and Inkeri Verkamo, A. (1997). Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1:259–289.

[Méger and Rigotti, 2004] Méger, N. and Rigotti, C. (2004). Constraint-based Mining of Episode Rules and Optimal Window Sizes. In *Proceedings of the European conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*.

[Mitchell, 1982] Mitchell, T. M. (1982). Generalization as Search. *Artificial Intelligence*, 18(2):203 – 226.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc.

[Moerchen, 2006a] Moerchen, F. (2006a). Algorithms for Time Series Knowledge Mining. In *Proceedings of the international conference on Knowledge Discovery and Data mining (SIGKDD)*.

[Moerchen, 2006b] Moerchen, F. (2006b). *Time Series Knowledge Mining*. PhD thesis, Philipps-University Marburg.

[Moerchen and Fradkin, 2010] Moerchen, F. and Fradkin, D. (2010). Robust Mining of Time Intervals with Semi-interval Partial Order Patterns. In *Proceedings of the SIAM international conference on Data Mining (SDM)*.

[Moerchen and Ultsch, 2005] Moerchen, F. and Ultsch, A. (2005). Optimizing Time Series Discretization for Knowledge Discovery. In *Proceedings of the international conference on Knowledge discovery in data mining (SIGKDD)*.

[Morishita and Sese, 2000] Morishita, S. and Sese, J. (2000). Tranversing Itemset Lattices with Statistical Metric Pruning. In *Proceedings of the symposium on Principles of database systems (PODS)*.

[Moskovitch and Shahar, 2009] Moskovitch, R. and Shahar, Y. (2009). Medical Temporal-Knowledge Discovery via Temporal Abstraction. In *Proceedings of the American Medical Informatics Association (AMIA)*.

[Nijssen et al., 2009] Nijssen, S., Guns, T., and De Raedt, L. (2009). Correlated Itemset Mining in ROC space: a Constraint Programming Approach. In *Proceedings of the international conference on Knowledge discovery and data mining (SIGKDD)*.

[Novak et al., 2009] Novak, P. K., Lavrač, N., and Webb, G. I. (2009). Supervised Descriptive Rule Discovery: A Unifying Survey of Contrast Set, Emerging Pattern and Subgroup Mining. *Journal of Machine Learning Research (JMLR)*, 10:377–403.

[Papadimitriou et al., 2005] Papadimitriou, S., Sun, J., and Faloutsos, C. (2005). Streaming Pattern Discovery in Multiple Time-Series. In *Proceedings of the international conference on Very Large Data Bases (VLDB)*.

[Papapetrou et al., 2005] Papapetrou, P., Kollios, G., Sclaroff, S., and Gunopulos, D. (2005). Discovering Frequent Arrangements of Temporal Intervals. In *Proceedings of the International Conference on Data Mining (ICDE)*.

[Pasquier et al., 1999] Pasquier, N., Bastide, Y., Taouil, R., and Lakhal, L. (1999). Discovering Frequent Closed Itemsets for Association Rules. In *Proceedings of the International Conference on Database Theory (ICDT)*.

[Patel et al., 2008] Patel, D., Hsu, W., and Lee, M. L. (2008). Mining Relationships among Interval-based Events for Classification. In *Proceedings of the international conference on Management Of Data (SIGMOD)*.

[Pei and Han, 2000] Pei, J. and Han, J. (2000). Can we push more Constraints into Frequent Pattern Mining? In *Proceedings of the international conference on Knowledge discovery and data mining (SIGKDD)*.

[Pei et al., 2001] Pei, J., Han, J., Mortazavi-asl, B., Pinto, H., Chen, Q., Dayal, U., and chun Hsu, M. (2001). PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In *Proceedings of the International Conference on Data Engineering (ICDE)*.

[Pei et al., 2007] Pei, J., Han, J., and Wang, W. (2007). Constraint-based Sequential Pattern Mining: the Pattern-growth Methods. *Journal of Intelligent Information Systems*, 28:133–160.

[Pendelton et al., 2006] Pendelton, R., Wheeler, M., and Rodgers, G. (2006). Argatroban Dosing of Patients with Heparin Induced Thrombocytopenia and an Alevated aPTT due to Antiphospholipid Antibody Syndrome. *The Annals of Pharmacotherapy*, 40:972Ű–976.

[Plamondon and Srihari, 2000] Plamondon, R. and Srihari, S. N. (2000). Online and Offline Handwriting Recognition: A Comprehensive Survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):63–84.

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of Decision Trees. In *Machine Learning*, pages 81–106.

[Quinlan, 1990] Quinlan, J. R. (1990). Learning Logical Definitions from Relations. *Machine Learning*, 5:239–266.

[Quinlan, 1993] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on Hidden Markov Models and selected applications in Speech Recognition. In *Proceedings of the IEEE*, pages 257–286.

[Ratanamahatana and Keogh, 2005] Ratanamahatana, C. and Keogh, E. J. (2005). Three Myths about Dynamic Time Warping Data Mining. In *Proceedings of the SIAM international conference on Data Mining (SDM)*.

[Rojas, 1996] Rojas, R. (1996). *Neural Networks: a systematic introduction*. Springer, 1st edition.

[Sacchi et al., 2007] Sacchi, L., Larizza, C., Combi, C., and Bellazzi, R. (2007). Data mining with Temporal Abstractions: learning rules from time series. *Data Mining and Knowledge Discovery*.

[Savasere et al., 1995] Savasere, A., Omiecinski, E., and Navathe, S. B. (1995). An Efficient Algorithm for Mining Association Rules in Large Databases. In *Proceedings of the international conference on Very Large Data Bases (VLDB)*.

[Schapire and Singer, 1999] Schapire, R. E. and Singer, Y. (1999). Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, pages 80–91.

[Sebastiani, 2002] Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*.

[Shahar, 1997] Shahar, Y. (1997). A Framework for Knowledge-Based Temporal Abstraction. *Artificial Intelligence, 90:79-133*.

[Shumway and Stoffer, 2006] Shumway, R. H. and Stoffer, D. S. (2006). *Time Series Analysis and Its Applications: With R Examples*. Springer, 2nd edition.

[Siebes et al., 2006] Siebes, A., Vreeken, J., and van Leeuwen, M. (2006). Item Sets that Compress. In *Proceedings of the SIAM international conference on Data Mining (SDM)*.

[Smyth and Goodman, 1992] Smyth, P. and Goodman, R. M. (1992). An Information Theoretic Approach to Rule Induction from Databases. *IEEE Transactions on Knowledge and Data Engineering*.

[Srikant and Agrawal, 1996] Srikant, R. and Agrawal, R. (1996). Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proceedings of the international conference on Extending Database Technology (EDBT)*.

[Srivastava et al., 2008] Srivastava, A., Kundu, A., Sural, S., and Majumdar, A. (2008). Credit Card Fraud Detection Using Hidden Markov Model. *IEEE Transactions on Dependable and Secure Computing*.

[Toivonen, 1996] Toivonen, H. (1996). Sampling Large Databases for Association Rules. In *Proceedings of the international conference on Very Large Data Bases (VLDB)*.

[Tseng and Lee, 2005] Tseng, V. S.-M. and Lee, C.-H. (2005). CBS: A New Classification Method by Using Sequential Patterns. In *Proceedings of the SIAM international conference on Data Mining (SDM)*.

[Vail et al., 2007] Vail, D. L., Veloso, M. M., and Lafferty, J. D. (2007). Conditional Random Fields for Activity Recognition. In *Proceedings of the international joint conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[Valko and Hauskrecht, 2010] Valko, M. and Hauskrecht, M. (2010). Feature Importance Analysis for Patient Management Decisions. In *Proceedings of medical informatics (MedInfo)*.

[Vanetik et al., 2002] Vanetik, N., Gudes, E., and Shimony, S. E. (2002). Computing Frequent Graph Patterns from Semistructured Data. In *Proceedings of the International Conference on Data Mining (ICDM)*.

[Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, NY.

[Veloso et al., 2006] Veloso, A., Meira Jr., W., and Zaki, M. J. (2006). Lazy Associative Classification. In *Proceedings of the International Conference on Data Mining (ICDM)*.

[Villafane et al., 2000] Villafane, R., Hua, K. A., Tran, D., and Maulik, B. (2000). Knowledge Discovery from Series of Interval Events. *Journal of Intelligent Information Systems*, 15:71–89.

[Wang and Han, 2004] Wang, J. and Han, J. (2004). BIDE: Efficient Mining of Frequent Closed Sequences. In *Proceedings of the International Conference on Data Engineering (ICDE)*.

[Wang et al., 2003a] Wang, J., Han, J., and Pei, J. (2003a). CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. In *Proceedings of the international conference on Knowledge Discovery and Data mining (SIGKDD)*.

[Wang and Karypis, 2005] Wang, J. and Karypis, G. (2005). HARMONY: Efficiently mining the best rules for classification. In *Proceedings of the SIAM international conference on Data Mining (SDM)*.

[Wang et al., 2003b] Wang, K., Jiang, Y., and Lakshmanan, L. V. S. (2003b). Mining Unexpected Rules by Pushing User Dynamics. In *Proceedings of the international conference on Knowledge Discovery and Data mining (SIGKDD)*.

[Warkentin, 2000] Warkentin, T. (2000). Heparin-induced thrombocytopenia: pathogenesis and management. *British Journal of Haematology*, 121:535–555.

[Webb, 2007] Webb, G. I. (2007). Discovering Significant Patterns. *Machine Learning*, 68(1):1–33.

[Weng and Shen, 2008] Weng, X. and Shen, J. (2008). Classification of Multivariate Time Series using two-dimensional Singular Value Decomposition. *Knowledge-Based Systems*, 21(7):535 – 539.

[Winarko and Roddick, 2007] Winarko, E. and Roddick, J. F. (2007). ARMADA - An Algorithm for Discovering Richer Relative Temporal Association Rules from Interval-based Data. *Data and Knowledge Engineering*, 63:76–90.

[Wong et al., 2005] Wong, W.-K., Moore, A. W., Cooper, G. F., and Wagner, M. M. (2005). What's Strange About Recent Events (WSARE): An Algorithm for the Early Detection of Disease Outbreaks. *Journal of Machine Learning Research*, 6:1961–1998.

[Wu and Chen, 2007] Wu, S.-Y. and Chen, Y.-L. (2007). Mining Nonambiguous Temporal Patterns for Interval-Based Events. *IEEE Transactions on Knowledge and Data Engineering*, 19:742–758.

[Xi et al., 2006] Xi, X., Keogh, E., Shelton, C., Wei, L., and Ratanamahatana, C. A. (2006). Fast Time Series Classification using Numerosity Reduction. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[Xin et al., 2006] Xin, D., Cheng, H., Yan, X., and Han, J. (2006). Extracting Redundancy-aware Top-k Patterns. In *Proceedings of the international conference on Knowledge discovery and data mining (SIGKDD)*.

[Xin et al., 2005] Xin, D., Han, J., Yan, X., and Cheng, H. (2005). Mining Compressed Frequent-pattern Sets. In *Proceedings of the 31st international conference on Very large data bases (VLDB)*.

[Xu et al., 2004] Xu, W., Guan, C., Siong, C., Sitaram, R., and Thulasidas, M. (2004). High Accuracy Classification of EEG Signal. In *International Conference of Pattern Recognition (ICPR)*.

[Yan et al., 2005] Yan, X., Cheng, H., Han, J., and Xin, D. (2005). Summarizing Itemset Patterns: a Profile-based Approach. In *Proceedings of the international conference on Knowledge discovery in data mining (SIGKDD)*.

[Yan and Han, 2002] Yan, X. and Han, J. (2002). gSpan: Graph-Based Substructure Pattern Mining. In *Proceedings of the International Conference on Data Mining (ICDM)*.

[Yan et al., 2003] Yan, X., Han, J., and Afshar, R. (2003). CloSpan: Mining Closed Sequential Patterns in Large Datasets. In *Proceedings of the SIAM international conference on Data Mining (SDM)*.

[Yang, 2004] Yang, G. (2004). The Complexity of Mining Maximal Frequent Itemsets and Maximal Frequent Patterns. In *Proceedings of the international conference on Knowledge Discovery and Data mining (SIGKDD)*.

[Yang and Shahabi, 2004] Yang, K. and Shahabi, C. (2004). A PCA-based Similarity Measure for Multivariate Time Series. In *Proceedings of the international workshop on Multimedia databases*.

[Yang et al., 2005] Yang, Y., Webb, G. I., and Wu, X. (2005). Discretization Methods. In *The Data Mining and Knowledge Discovery Handbook*, pages 113–130. Springer.

[Ye and Keogh, 2009] Ye, L. and Keogh, E. (2009). Time Series Shapelets: A New Primitive for Data Mining. In *Proceedings of the international conference on Knowledge Discovery and Data mining (SIGKDD)*.

[Yin and Han, 2003] Yin, X. and Han, J. (2003). CPAR: Classification based on Predictive Association Rules. In *Proceedings of the SIAM international conference on Data Mining (SDM)*.

[Yu et al., 2012] Yu, K., Ding, W., Simovici, D. A., and Wu, X. (2012). Mining Emerging Patterns by Streaming Feature Selection. In *Proceedings of the international conference on Knowledge Discovery and Data mining (SIGKDD)*.

[Zaki, 2000] Zaki, M. J. (2000). Scalable Algorithms for Association Mining. *IEEE Transaction on Knowledge and Data Engineering*, 12:372–390.

[Zaki, 2001] Zaki, M. J. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning*, 42:31–60.

[Zaki and Hsiao, 2002] Zaki, M. J. and Hsiao, C.-J. (2002). CHARM: An Efficient Algorithm for Closed Itemset Mining. In *Proceedings of the SIAM international conference on Data Mining (SDM)*.