Knowl Inf Syst (2016) 46:115–150
DOI 10.1007/s10115-015-0819-6

REGULAR PAPER

# An efficient pattern mining approach for event detection in multivariate temporal data

**Iyad Batal · Gregory F. Cooper · Dmitriy Fradkin · James Harrison Jr. · Fabian Moerchen · Milos Hauskrecht**

Received: 30 November 2013 / Revised: 31 August 2014 / Accepted: 6 December 2014 /
Published online: 21 January 2015

**Abstract** This work proposes a pattern mining approach to learn event detection models from complex multivariate temporal data, such as electronic health records. We present recent temporal pattern mining, a novel approach for efficiently finding predictive patterns for event detection problems. This approach first converts the time series data into time-interval sequences of temporal abstractions. It then constructs more complex time-interval patterns backward in time using temporal operators. We also present the minimal predictive recent temporal patterns framework for selecting a small set of predictive and non-spurious patterns. We apply our methods for predicting adverse medical events in real-world clinical

I. Batal (✉)
GE Global Research, San Ramon, CA, USA
e-mail: iyadbatal@gmail.com; iyad.batal@ge.com

G. F. Cooper
Department of Biomedical Informatics, University of Pittsburgh,
Pittsburgh, PA, USA
e-mail: gfc@pitt.edu

D. Fradkin
Siemens Corporate Research, Princeton, NJ, USA
e-mail: dmitriy.fradkin@siemens.com

J. Harrison Jr.
Department of Public Health Sciences, University of Virginia,
Charlottesville, VA, USA
e-mail: james.harrison@virginia.edu

F. Moerchen
Amazon, Seattle, WA, USA
e-mail: moerchen@amazon.com

M. Hauskrecht
Department of Computer Science, University of Pittsburgh,
Pittsburgh, PA, USA
e-mail: milos@cs.pitt.edu

data. The results demonstrate the benefits of our methods in learning accurate event detection models, which is a key step for developing intelligent patient monitoring and decision support systems.

**Keywords** Temporal data mining · Electronic health records · Temporal abstractions · Time-interval patterns · Recent temporal patterns · Event detection

## 1 Introduction

Advances in data collection and data storage technologies have led to the emergence of complex multivariate temporal datasets, where the data instances are traces of complex behaviors characterized by time series of multiple variables. Designing algorithms capable of analyzing and learning from such data is one of the most challenging topics of data mining research.

In this work, we study the supervised learning task of event detection in multivariate temporal data. In this task, the labels denote different events that are associated with specific time points during the temporal instances. The goal is to learn how to accurately detect (in time) the occurrence of these events for future instances (a monitoring task). This setting appears in a wide range of applications, such as the detection of adverse medical events (e.g., onsets of diseases or drug toxicity) in clinical data [18], the detection of equipment malfunctions [16], fraud detection [42], environmental monitoring [31], intrusion detection [11] and others.

Given that the events are associated with specific time points during the instances, the temporal observations that are close to the time of the event are typically the most important for prediction. Consequently, the context for prediction is often local and affected by the most recent behavior of the monitored instance. The focus of this paper is to develop a temporal pattern mining method that takes into account the local nature of decisions for monitoring and event detection problems.

We primarily focus on data in electronic health record (EHR) systems. In these data, each record (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. The task is to learn how to predict these adverse medical events for future patients. This is practically very useful for intelligent patient monitoring, outcome prediction and clinical decision support.

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 time series classification methods [10,43], time series similarity measures [38,51] and time series feature extraction methods [7,25,46] cannot be directly applied to the raw EHR data.

In this paper, we present a pattern mining approach that can handle complex temporal data such as EHR. The first step is to define a pattern language that can adequately represent the temporal dimension of the data. To do this, we rely on temporal abstractions [40] to convert time series variables into *time-interval* sequences of abstract states and on Höppner's representation of time-interval patterns [20,21] to define complex temporal interactions among multiple states. For example, this allows us to express a concept like "the administration of heparin precedes a decreasing trend in platelet counts."

After defining the pattern language, we need to design an efficient algorithm for finding useful patterns in time-interval (temporally abstracted) data. Unlike the existing methods that mine all frequent time-interval patterns in an unsupervised setting [20,22,28,32,39,47,48], we are only interested in mining those patterns that are important for the event detection task (patterns that describe the temporal behavior that led to the development of the event).

To achieve this goal, we propose *Recent Temporal Pattern* (***RTP***) mining, a novel approach that mines frequent time-interval patterns backward in time starting from patterns related to the most recent observations, which we argue are typically the most important for prediction. Applying this technique, temporal patterns that extend far into the past are likely to have low support in the data and hence would not be considered for prediction. 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.

In addition, we present the concept of *minimal predictive recent temporal patterns* (***MPRTP***), a pattern selection technique for choosing a small set of predictive and non-redundant RTPs. MPRTP applies Bayesian inference to evaluate the predictiveness of each RTP. Moreover, it considers the structure of patterns to ensure that every RTP in the result is significantly more predictive than all of its simplifications, which we demonstrate can eliminate a lot of spurious and redundant RTPs.

We test and demonstrate the usefulness of our methods on two real-world EHR datasets. The first dataset is a large-scale data that contains health records for 13,558 diabetic patients. The task is to predict and diagnose various types of disorders that are frequently associated with diabetes, such as cardiological, renal or neurological disorders. The second dataset contains records of post- cardiac surgical patients [17,18], and the task is to predict patients who are at risk of developing heparin-induced thrombocytopenia (HIT), a life-threatening condition that may develop in patients treated with heparin.

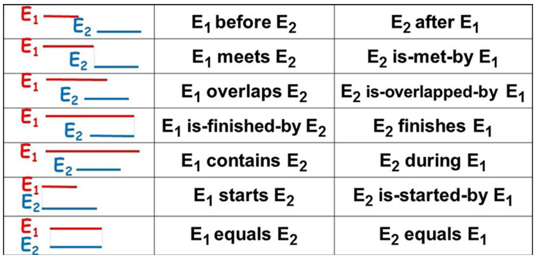
Our main contributions are summarized as follows:

- We propose recent temporal pattern (RTP) mining to find predictive time-interval patterns for event detection problems.
- We present an algorithm that mines frequent RTPs backward in time and show that it is much more scalable than mining all frequent time-interval patterns.
- We propose the concept of minimal predictive recent temporal patterns (MPRTP) for selecting predictive and non-spurious patterns.
- We present a mining algorithm which integrates MPRTP evaluation with frequent RTP mining and applies efficient pruning techniques to speed up the mining.
- We apply our methods to two real-world clinical datasets and show that they are able to efficiently find useful patterns for detecting adverse medical events. Furthermore, we show that the mined patterns are easy to interpret by domain experts, which is beneficial for knowledge discovery purposes.

*Roadmap* The rest of the paper is organized as follows. Section 2 outlines related research. Section 3 describes the background on temporal abstraction and temporal patterns for time-interval data. Section 4 describes the RTPs framework and explains our algorithm for mining frequent RTPs. Section 5 describes the MPRTPs framework and explains our mining algorithm. Section 6 presents the experimental evaluation. Finally, Sect. 7 concludes the paper.

## 2 Related research

This paper studies the problem of mining predictive time-interval patterns (defined over temporal abstractions) for event detection. The problem of mining time-interval patterns is a relatively young research field. Most of the techniques extend methods for mining sequential



| | | |
|---|---|---|
| | $E_1$ 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 $E_1$ |

**Fig. 1** Allen's temporal relations

patterns (time-point patterns) [3,35,50,53] to deal with the more complex case of time-interval data.[1]

In Allen [4], a logic on time intervals was formalized by specifying 13 possible temporal relations and showing their completeness. These relations are shown in Fig. 1. Allen's temporal relations have been used by the majority of research on mining time-interval data [20,22,30,32,33,47].

Kam and Fu [22] were the first to propose using Allen's relations for defining time-interval patterns (patterns consisting of multiple states). However, their representation was shown to be ambiguous (the same situation in the data can be described using several different patterns). Höppner [20] was the first to propose a non-ambiguous representation of time-interval patterns. The idea is to represent the pattern in the normalized form and to explicitly specify the temporal relations between all pairs of states. Höppner's representation has been later used by several research papers [30,32,47], and we adopt it as well in our work. Moerchen [28] proposed the time series knowledge representation (TSKR) as an alternative to using Allen's relations. TSKR is based on the concept of cords, which describe coincidence of several states. Another representation appeared in [48], where a time-interval pattern is represented as a sequence of the boundaries of its states. Although this representation is conceptually different from Höppner's representation, it can be shown that there is a one-to-one correspondence between them.

The related work on mining time-interval patterns [20,22,28,30,32,39,47,48] have been mostly applied in an unsupervised setting for mining all frequent patterns. On the other hand, our work studies the supervised event detection task for mining the most predictive patterns, which we achieve by incorporating the concept of recency in the mining process.

## 3 Preliminaries and definitions

In this section, we describe temporal abstraction and define temporal patterns for time-interval data. For temporal abstraction, we use trend and value abstractions. For the pattern language, we use Höppner's representation of time-interval patterns [20,21] with a simplified set of Allen's temporal relations.

### 3.1 Temporal abstraction

Temporal abstraction takes a numeric time series (e.g., series of creatinine values) and produces a sequence of abstractions $\langle v_1[s_1, e_1], \ldots, v_n[s_n, e_n]\rangle$, where $v_i \in \Sigma$ is an abstraction

[1] Sequential pattern mining is a special case of time-interval pattern mining, in which all intervals are simply time points with zero durations.

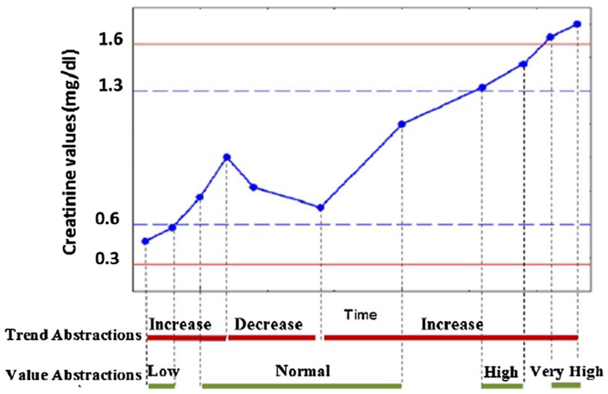


**Fig. 2** An example illustrating trend abstractions and value abstractions. The *blue dashed lines* represent the 25th and 75th percentiles of the values, and the *red solid lines* represent the 10th and 90th percentiles of the values (color figure online)

that holds from time $s_i$ to time $e_i$ and $\Sigma$ is the abstraction alphabet (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. We use the following abstractions: *Decreasing* ($D$), *Steady* ($S$) and *Increasing* ($I$), i.e., $\Sigma = \{D, S, I\}$. We obtain these abstractions by applying the sliding window segmentation method [24] and labeling the states according to the slopes of the fitted segments. For more information about trend segmentation, see [24].
2. *Value abstractions* segment the time series based on its values. We use the following abstractions: *Very Low* ($VL$), *Low* ($L$), *Normal* ($N$), *High* ($H$) and *Very High* ($VH$), i.e., $\Sigma = \{VL, L, N, H, VH\}$. We obtain these abstractions using the 10th, 25th, 75th and 90th percentiles on the laboratory values. That is, 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 2 shows the trend and value abstractions on a time series of creatinine values.

## 3.2 Multivariate state sequences

Let a **state** be an *abstraction value for a specific variable*. We denote a state $S$ by a pair $(F, V)$, where $F$ is a temporal variable (e.g., creatinine) and $V \in \Sigma$ is an abstraction value. 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 $(glucose, H, 5, 10)$ 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.

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

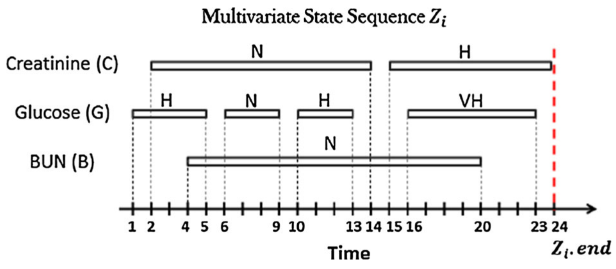


**Fig. 3** An MSS representing 24 days of a patient record. In this example, there are three temporal variables (creatinine, glucose and BUN) that are abstracted using value abstractions

**Definition 1** A **multivariate state sequence (MSS)** is represented as a series of state intervals that are ordered according to their start times:[3]

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

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 $\boldsymbol{Z.end}$ denote the end time of the MSS.

*Example 1* Figure 3 shows an MSS ($Z$) with three temporal variables: creatinine (C), glucose (G) and blood urea nitrogen (BUN) (B), where the variables are abstracted using value abstractions: $\Sigma = \{VL, L, N, H, VH\}$. Assuming the time granularity is days, this MSS represents 24 days of the patient's record ($Z.end = 24$). For instance, we can see that creatinine values are normal from day 2 until day 14, then become high from day 15 until day 24. We represent this MSS as follows: $\langle E_1 = (G, H, 1, 5), E_2 = (C, N, 2, 14), E_3 = (B, N, 4, 20), E_4 = (G, N, 6, 9), E_5 = (G, H, 10, 13), E_6 = (C, H, 15, 24), E_7 = (G, VH, 16, 23)\rangle$.

## 3.3 Temporal relations

Most of Allen's relations require equality of one or two of the interval boundaries. That is, there is only a slight difference between *overlaps*, *is-finished-by*, *contains*, *starts* and *equals* relations (see Fig. 1). Using all Allen's relations can be problematic when the time information in the data is *noisy* (not very precise), which is the case for EHR data. The reason is that this leads to the problem of *pattern fragmentation* [29], which means that we obtain many different temporal patterns that describe a very similar concept in the data. Furthermore, if we use all Allen's relations, the search space of temporal patterns becomes extremely large.

To alleviate the above-mentioned problems, 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 $E_j$:

- $E_i$ is ***before*** $E_j$, denoted as $\mathbf{b}(\boldsymbol{E_i}, \boldsymbol{E_j})$, if $E_i.e < E_j.s$ ($E_i$ finishes before $E_j$ starts), which is the same as Allen's *before*.
- $E_i$ ***co-occurs*** with $E_j$, denoted as $\mathbf{c}(\boldsymbol{E_i}, \boldsymbol{E_j})$, if $E_i.s \leq E_j.s \leq E_i.e$ ($E_i$ 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* (see Fig. 1).

[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 of their variable names (as proposed by [21]).

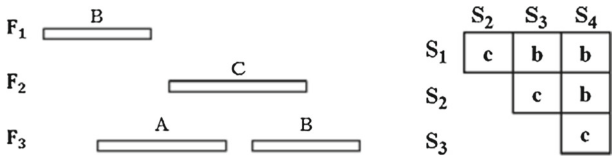


**Fig. 4** A temporal pattern with states $\langle (F_1, B), (F_3, A), (F_2, C), (F_3, B) \rangle$ and temporal relations $R_{1,2} = \mathbf{c}$, $R_{1,3} = \mathbf{b}$, $R_{1,4} = \mathbf{b}$, $R_{2,3} = \mathbf{c}$, $R_{2,4} = \mathbf{b}$ and $R_{3,4} = \mathbf{c}$

## 3.4 Temporal patterns

In order to define temporal patterns for abstracted time-interval data (time-interval patterns), we combine basic states (abstractions for specific variables) using temporal relations. We adopt Höppner's representation of time-interval patterns [20], where the permitted temporal relations are either *before* (**b**) or *co-occurs* (**c**) (defined above). From hereafter, we use the terms "temporal pattern" and "time-interval pattern" interchangeably.

**Definition 2** A **temporal pattern** is defined as $P = (\langle S_1, \ldots, S_k \rangle, R)$ 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:

$i \in \{1, \ldots, k-1\} \wedge j \in \{i+1, \ldots, k\} : R_{i,j} \in \{\mathbf{b}, \mathbf{c}\}$ 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 4 shows a graphical representation of a 4-*pattern* with states: $S_1 = (F_1, B)$, $S_2 = (F_3, A)$, $S_3 = (F_2, C)$ and $S_4 = (F_3, B)$. These states are abstractions of temporal variables $F_1$, $F_2$ and $F_3$ using abstraction alphabet $\Sigma = \{A, B, C\}$ (the abstractions in $\Sigma$ can be trend abstractions, value abstractions or any other type). The half matrix on the right represents the temporal relations between every state and the states that follow it. For example, $R_{2,3} = \mathbf{c}$ means that the second state $S_2 = (F_3, A)$ *co-occurs* with the third state $S_3 = (F_2, C)$.

**Definition 3** Given an MSS $Z = \langle E_1, E_2, \ldots, E_l \rangle$ and a temporal pattern $P = (\langle S_1, \ldots, S_k \rangle, R)$, we say that $Z$ **contains** $P$, denoted as $\boldsymbol{P \in Z}$, if there is an injective mapping $\pi$ that matches every state $S_i$ in $P$ to a state interval $E_{\pi(i)}$ in $Z$ such that:

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

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.

*Example 2* Let $Z$ be the MSS (abstracted EHR) in Fig. 3 and let $P$ be temporal pattern $(\langle (G, H), (C, N), (G, VH) \rangle,\; R_{1,2} = \mathbf{c},\; R_{1,3} = \mathbf{b}, R_{2,3} = \mathbf{b})$, i.e., Glucose=High co-occurs with Creatinine=Normal and both of them are before glucose=very high. $P$ is contained in $Z$ ($P \in Z$) because every state of $P$ can be matched with a state interval in $Z$ and these state intervals satisfy all the relations specified by $P$.

## 4 Mining recent temporal patterns

In this section, we first introduce the concept of *recent temporal patterns* (RTPs) and illustrate their properties. After that, we describe an efficient algorithm for mining frequent RTPs.

### 4.1 Recent temporal patterns

In this work, we study the problem of mining predictive temporal patterns for event detection. In this setting, each training instance $x_i$ is a multivariate temporal instance up to time $t_i$ (e.g., the first 5 days of the health record for a specific patient) and it is associated with a label that indicates whether or not the event of interest (e.g., a specific disorder) occurred at time $t_i$. The objective to learn temporal patterns that capture the behavior that led to the development of the event and apply these patterns to detect the occurrence of events in future instances (e.g., monitor new patients).

For event detection problems, recent observations of the temporal variables of $x_i$ (observations close to $t_i$) are typically the most important for predicting the label. For example, in the clinical domain, recent laboratory measurements are usually more indicative of the patient's current health status (hence more predictive) than old laboratory measurements. For instance, to diagnose a disease, the physician usually first looks at the most recent data and then looks back in time to understand how the data evolved over time. Our objective is to develop a temporal pattern mining method that takes into account this local nature of decisions for monitoring and event detection problems. We start by defining the concept of recent temporal patterns and then present an efficient algorithm to automatically mine these patterns from data.[4]

**Definition 4** Given an MSS $Z = \langle E_1, E_2, \ldots, E_l \rangle$ 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 for its variable (there is no state interval for the variable of $E_j$ that appears after $E_j$). Note that if the maximum gap $g = \infty$, any state interval of $Z$ ($E_j \in Z$) would be a recent state interval.

**Definition 5** Given an MSS $Z = \langle E_1, E_2, \ldots, E_l \rangle$ and a maximum gap parameter $g$, we say that temporal pattern $P = (\langle S_1, \ldots, S_k \rangle, R)$ is a **recent temporal pattern (RTP)** for $Z$, denoted as $\boldsymbol{R_g(P, Z)}$, if **all** of the following conditions are satisfied:

1. $P \in Z$ (Definition 3) with a mapping $\pi$ 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. $\forall i \in \{1, \ldots, k-1\}$, $S_i$ and $S_{i+1}$ match state intervals that are no more than $g$ time units 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 for MSS $Z$, (1) $P$ should be contained in $Z$ (Definition 3), (2) the last state of $P$ should map to a recent state

[4] This section contains materials that have been published in [6].

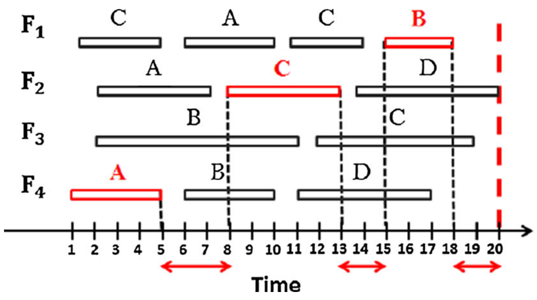


**Fig. 5** An MSS with four temporal variables ($F_1$, $F_2$, $F_3$ and $F_4$) that are abstracted using abstraction alphabet $\Sigma = \{A, B, C, D\}$. Temporal pattern $(\langle(F_4, A), (F_2, C), (F_1, B)\rangle,\ R_{1,2}{=}\mathbf{b}, R_{1,3}{=}\mathbf{b}, R_{2,3} = \mathbf{b})$ is an RTP for this MSS when the maximum gap parameter $g \geq 3$ (color figure online)

interval in $Z$ (Definition 4) and (3) any pair of consecutive states in $P$ should map to state intervals that are "close to each other" (within the maximum gap $g$). 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 for $Z$. Let us denote an RTP that contains $k$ states as a ***k*-RTP**.

*Example 3* Let $Z$ be the MSS in Fig. 5, which has four temporal variables ($F_1$ to $F_4$) that are abstracted using alphabet $\Sigma = \{A, B, C, D\}$. Let the maximum gap parameter be $g = 3$. Temporal pattern $P = (\langle(F_4, A),\ (F_2, C),\ (F_1, B)\rangle,\ R_{1,2} = \mathbf{b},\ R_{1,3} = \mathbf{b},\ R_{2,3} = \mathbf{b})$ is an RTP for $Z$ because $P \in Z$ (shown in red in Fig. 5), $(F_1, B, 15, 18)$ 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" $(F_2, C, 8, 13)(8 - 5 \leq g)$.

*Example 4* Let $Z$ be the MSS (abstracted EHR) in Fig. 3 and let $g = 3$ days. Temporal pattern $P_1 = (B, N)\ c\ (C, H)$ (i.e., BUN=Normal co-occurs with Creatinine=High) is an RTP in $Z$. On the other hand, $P_2 = (G, H)\ b\ (G, N)$ (i.e., Glucose=High before Glucose=Normal) is contained in $Z$: $P_2 \in Z$, but it is not an RTP because the second condition of Definition 5 is violated: $(G, N, 6, 9)$ is not a recent state interval.

**Definition 6** Given temporal patterns $P = (\langle S_1, \ldots, S_{k_1}\rangle, R)$ and $P' = (\langle S'_1, \ldots, S'_{k_2}\rangle, R')$ with $k_1 < k_2$, we say that $P$ is a **suffix subpattern** of $P'$, denoted as **Suffix**$(\boldsymbol{P}, \boldsymbol{P'})$, if

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

This definition simply says that for a $k_1$-*pattern* $P$ to be a suffix subpattern of a $k_2$-*pattern* $P'$ ($k_1 < k_2$), $P$ should match the last $k_1$ states of $P'$ and should satisfy among them the same temporal relations that are satisfied in $P'$. For example, pattern $(\langle(F_4, A),\ (F_2, C),\ (F_3, B)\rangle,\ R_{1,2} = \mathbf{c},\ R_{1,3} = \mathbf{b},\ R_{2,3} = \mathbf{c})$ is a suffix subpattern of the pattern in Fig. 4.

If $P$ is a suffix subpattern of $P'$, we say that $P'$ is a **backward extension superpattern** of $P$, which we abbreviate as ***b-extension***.

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

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

The proof follows directly from the Definition of RTP.

*Example 5* In Example 3, we showed that pattern $(\langle (F_4, A),\ (F_2, C),\ (F_1, B)\rangle,\ R_{1,2} = \mathbf{b},\ R_{1,3} = \mathbf{b},\ R_{2,3} = \mathbf{b})$ is an RTP for MSS $Z$ in Fig. 5 when the maximum gap is 3. Proposition 1 says that its suffix subpattern $(\langle (F_2, C),\ (F_1, B)\rangle,\ R_{1,2} = \mathbf{b})$ must also be an RTP for $Z$. However, this does not imply that subpattern $(\langle (F_4, A),\ (F_2, C)\rangle, R_{1,2} = \mathbf{b})$ must be an RTP (the second condition of Definition 5 is violated) nor that subpattern $(\langle (F_4, A),\ (F_1, B)\rangle, R_{1,2} = \mathbf{b})$ must be an RTP (the third condition of Definition 5 is violated).

One of the most used statistics in pattern mining is the support of the pattern [2] that counts the occurrences of the pattern in data. We define the support for RTPs as follows:

**Definition 7** Given a dataset $D$ of MSS and a maximum gap parameter $g$, the support of RTP $P$ in $D$ is defined as $\textbf{RTP-sup}_{\boldsymbol{g}}\boldsymbol{(P, D)} = |\{Z_i : Z_i \in D \wedge R_g(P, Z_i)\}|$.

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

A key property that all frequent pattern mining algorithms rely on is the Apriori property [2], which states that the support of a pattern is no more than the support of any of its subpatterns. This property holds for RTPs on suffix subpatterns. In particular, Proposition 1 implies that the *RTP-sup* of an RTP cannot be larger than the *RTP-sup* of its suffix subpatterns:

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

This corollary will be used by our algorithm for mining frequent RTPs.

## 4.2 The mining algorithm

In this section, we present the algorithm for mining frequent RTPs. In order to improve the efficiency and effectiveness of the mining algorithm, we utilize the label information and mine frequent RTPs for each class label separately using a *local minimum support* that is related to the size of the population of that label. For example, suppose we have 10K instances, 1K of them has the event of interest (positives) and 9K do not have the event (negatives). If we set the local minimum support to be 10 % the number of instances, we mine frequent RTPs from the positives and the negatives separately using $\sigma_{\text{pos}} = 100$ for the positives and $\sigma_{\text{neg}} = 900$ for the negatives. This approach is more appropriate when mining pattern in a supervised setting compared to mining frequent patterns from the entire data using a single global minimum support for the following reasons:

1. Mining patterns that are frequent for one of the classes (hence potentially predictive for that class) is more efficient than mining patterns that are globally frequent.[5]
2. For unbalanced data, mining frequent patterns using a global minimum support may result in missing many important patterns for the rare class.

The algorithm for mining frequent RTPs for class $y$ takes as input $D_y$: the MSS from $y$, $g$: the maximum gap parameter and $\sigma_y$: the local minimum support threshold for $y$. It outputs all temporal patterns that have an *RTP-sup* (Definition 7) in $D_y$ that is larger or equal to $\sigma_y$:

$$\left\{P_1, \ldots, P_m : \textit{RTP-sup}_g(P_i, D_y) \geq \sigma_y\right\}$$

[5] It is more efficient to mine patterns that cover more than $n$ instances in one of the classes compared to mining patterns that cover more than $n$ instances in the entire database (the former is always a subset of the latter).

The algorithm explores the space of temporal patterns level by level. It first finds all frequent 1-*RTPs* (RTPs that consist of a single state). Then, it extends these patterns ***backward in time*** to find more complex patterns (frequent 2-*RTPs*, then frequent 3-*RTPs* and so on). For each level $k$, the algorithm performs the following two steps to obtain the frequent $(k+1)$-*RTPs*:

1. **Candidate generation:** Generate candidate $(k+1)$-*patterns* by extending frequent $k$-*RTPs* backward in time.
2. **Counting:** Obtain the frequent $(k+1)$-*RTPs* by removing the candidates with *RTP-sup* less than $\sigma_y$.

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

In the following, we describe in detail the candidate generation algorithm. Then, we propose techniques to improve the efficiency of both candidate generation and counting.

### *4.2.1 Backward candidate generation*

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

In order to fully define a candidate, we still need to specify the temporal relations between the new state $S'_1$ and states $S'_2, \ldots, S'_{k+1}$, i.e., we should define $R'_{1,i}$ for $i \in \{2, \ldots, k+1\}$. 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 results.

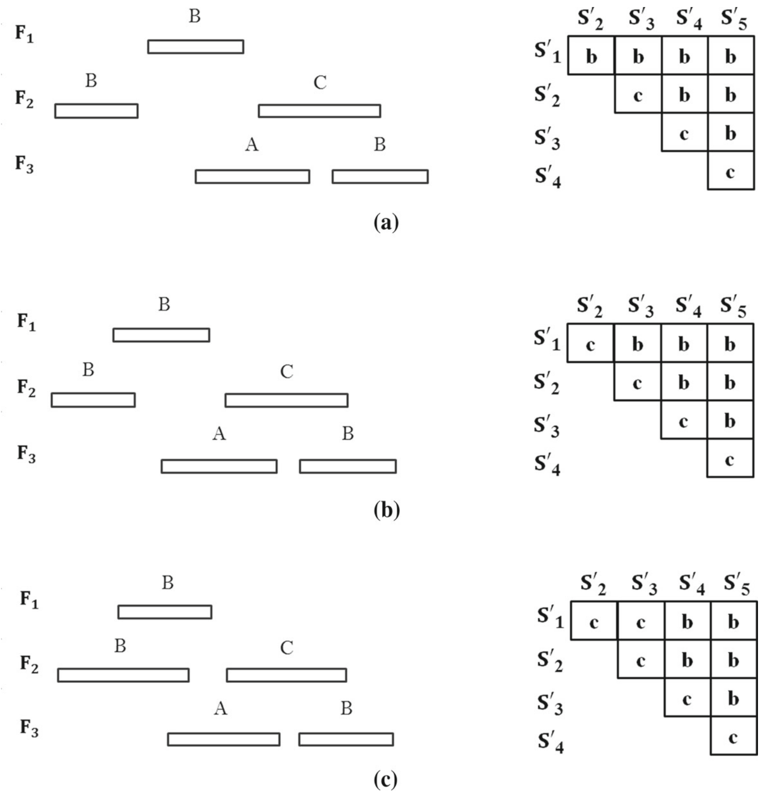
### *4.2.2 Improving the efficiency of candidate generation*

**Definition 8** 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 (they are invalid patterns). We introduce the following two lemmas to avoid generating incoherent candidates when specifying the relations $R'_{1,i} : i \in \{2, \ldots, k+1\}$ in candidates of the form $P' = (\langle S'_1, \ldots, S'_{k+1} \rangle, R')$. The first lemma restricts the patterns that include the same variable with two co-occurring states. The second lemma restricts the patterns that would violate the ordering of relations in time.

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

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



(a)

| | $S'_2$ | $S'_3$ | $S'_4$ | $S'_5$ |
|---|---|---|---|---|
| $S'_1$ | b | b | b | b |
| $S'_2$ | | c | b | b |
| $S'_3$ | | | c | b |
| $S'_4$ | | | | c |

(b)

| | $S'_2$ | $S'_3$ | $S'_4$ | $S'_5$ |
|---|---|---|---|---|
| $S'_1$ | c | b | b | b |
| $S'_2$ | | c | b | b |
| $S'_3$ | | | c | b |
| $S'_4$ | | | | c |

(c)

| | $S'_2$ | $S'_3$ | $S'_4$ | $S'_5$ |
|---|---|---|---|---|
| $S'_1$ | c | c | b | b |
| $S'_2$ | | c | b | b |
| $S'_3$ | | | c | b |
| $S'_4$ | | | | c |

**Fig. 6** Coherent candidates that result from extending the temporal pattern in Fig. 4 backward with state $(F_2, B)$

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

*Proof* Assume that there exists an MSS $Z = \langle E_1, \ldots, E_l\rangle$ where $P' \in Z$. Let $\pi$ be the mapping from the states of $P'$ to the state intervals of $Z$. The definition of temporal patterns (Definition 2) and the fact that state intervals in $Z$ are ordered by their start values (Definition 1) implies that the matching state intervals $\langle E_{\pi(1)}, \ldots, E_{\pi(k+1)}\rangle$ are also ordered by their start times: $E_{\pi(1)}.s \le \cdots \le E_{\pi(k+1)}.s$. Hence, $E_{\pi(j)}.s \ge E_{\pi(i)}.s$ since $j > i$. We also know that $E_{\pi(1)}.e < E_{\pi(i)}.s$ because $R'_{1,i} = \mathbf{b}$. Therefore, $E_{\pi(1)}.e < E_{\pi(j)}.s$. However, since $R'_{1,j} = \mathbf{c}$, then $E_{\pi(1)}.e \ge E_{\pi(j)}.s$, which is a contradiction. Therefore, there is no MSS that contains $P'$. □

*Example 6* Assume we want to extend $P = (\langle S_1 = (F_1, B), S_2 = (F_3, A), S_3 = (F_2, C), S_4 = (F_3, B)\rangle, R_{1,2} = \mathbf{c}, R_{1,3} = \mathbf{b}, R_{1,4} = \mathbf{b}, R_{2,3} = \mathbf{c}, R_{2,4} = \mathbf{b}, R_{3,4} = \mathbf{c})$ in Fig. 4 with state $S_{new} = (F_2, B)$ to generate candidates of the form $(\langle S'_1 = (F_2, B),\ S'_2 = (F_1, B), S'_3 = (F_3, A), S'_4 = (F_2, C),\ S'_5 = (F_3, B)\rangle, R')$. First, we set the relations in $R'$ for the last four states ($S'_2$, $S'_3$, $S'_4$ and $S'_5$) to be the same as defined by $R$. Now, we have to specify relations $R'_{1,i}$ for $i \in \{2, \ldots, k+1\}$. $R'_{1,2}$ is allowed to be either before ($R'_{1,2} = \mathbf{b}$) or co-occurs ($R'_{1,2} = \mathbf{c}$). If $R'_{1,2} = \mathbf{b}$, then all the following relations must be before according to Lemma 2, resulting in the candidate shown in Fig. 6a. If $R'_{1,2} = \mathbf{c}$, then $R'_{1,3}$ is allowed to be either before ($R'_{1,3} = \mathbf{b}$) or co-occurs ($R'_{1,3} = \mathbf{c}$), resulting in the candidates shown in Fig. 6b, c, respectively. Now, according to Lemma 1, $R'_{1,4} \neq \mathbf{c}$ because both $S'_1$ and $S'_4$ belong to the same temporal variable (variable $F_2$). As we see, we have reduced the number of candidates that result from adding state $(F_2, B)$ to a 4-*RTP* $P$ from $2^4 = 16$ in the naïve way to only 3.

The following theorem gives the bound on the number of possible extensions of an RTP with a new state (an abstraction value for a specific variable).

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

*Proof* We know that every candidate $P' = (\langle S'_1, \ldots, S'_{k+1}\rangle, R')$ corresponds to a specific assignment of $R'_{1,i} \in \{\mathbf{b}, \mathbf{c}\}$ for $i \in \{2, \ldots, k+1\}$. 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 some point $q \in \{2, \ldots, k+1\}$ in the pattern, all subsequent relations ($i > q$) should be *before* as well:

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

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, $R'_{1,j} \neq \mathbf{c}$. 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: $P = (\langle S_1, \ldots, S_k\rangle, R)$; a new state: $S_{new}$
Output: Coherent candidates: $C$
1  $S'_1 = S_{new}$; $S'_{i+1} = S_i : i \in \{1, \ldots, k\}$;
2  $R'_{i+1,j+1} = R_{i,j} : i \in \{1, \ldots, k-1\}, j \in \{i+1, \ldots, k\}$;
3  $R'_{1,i} = b : i \in \{2, \ldots, k+1\}$;     $P' = (\langle S'_1, \ldots, S'_{k+1}\rangle, R')$;
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          $R'_{1,i} = c$;     $P' = (\langle S'_1, \ldots, S'_{k+1}\rangle, R')$;
10         $C = C \cup \{P'\}$;
11     end
12 end
13 return $C$
```

The bound on the number of possible extensions of a *k-RTP* with a new state can be used to calculate the bound on the total number of possible extensions of size $k + 1$.

**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 no more than $(k+1) \times |L| \times |\mathbf{F_k}|$.*

### *4.2.3 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 [52] that introduced the *vertical data format* for itemset mining and later applied it for sequential pattern mining [53].

Let us associate every frequent RTP $P$ with a list of identifiers of all MSS in $D_y$ for which $P$ is an RTP (Definition 5):

$$P.RTP\text{-}list = \langle i_1, i_2, \ldots, i_n \rangle : Z_{i_j} \in D_y \wedge R_g(P, Z_{i_j}) \quad \text{for } j = 1, \ldots, n.$$

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 3):

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

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** *If $P'$ be a b-extension of RTP $P$ with state $S$ and $P'.p - RTP\text{-}list = P.RTP\text{-}list \ \cap \ S.list$, then $P'.RTP\text{-}list \subseteq P'.p\text{-}RTP\text{-}list$.*

*Proof* Assume $Z_i$ is an MSS for which $P'$ is an RTP: $R_g(P', Z_i)$. By definition, $i \in P'.RTP\text{-}list$. We know that $R_g(P', Z_i) \implies P' \in Z_i \implies S \in Z_i \implies i \in S.list$. Also, we know that $Suffix(P, P') \implies R_g(P, Z_i)$ (according to Proposition 1) $\implies i \in P.RTP\text{-}list$. Therefore, $i \in P.RTP\text{-}list \cap S.list = P'.p\text{-}RTP\text{-}list$. □

Putting it all together, we compute the *RTP-lists* in the *counting phase* (based on the true matches) and the *p-RTP-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-RTP-list* because according to Proposition 2: $i \notin P'.p\text{-}RTP\text{-}list \implies i \notin P'.RTP\text{-}list \implies P'$ is not an RTP for $Z_i$. This offers a lot of computational savings because the *p-RTP-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 according to Algorithm 1. Line 6 computes the *p-RTP-list* for each candidate. Note that the cost of the intersection is *linear* in the length of the lists because the lists are always sorted according to the order of the instances in the data. Line 7 applies *additional pruning* to remove candidates that are guaranteed not to be frequent according to the following implication of Proposition 2:

$$|P'.p\text{-}RTP\text{-}list| < \sigma_y \implies |P'.RTP\text{-}list| = RTP\text{-}sup_g(P', D_y) < \sigma_y$$
$$\implies P' \text{ is not frequent}$$

**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 $p\text{-}RTP\text{-}lists$
1  $Cand = \Phi$;
2  foreach $P \in \mathbf{F_k}$ do
3      foreach $S \in L$ do
4          $C = extend\_backward(P, S)$;      (Algorithm 1)
5          for $q = 1$ to $| C |$ do
6              $C[q].p\text{-}RTP\text{-}list = P.RTP\text{-}list \cap S.list$;
7              if ( $| C[q].p\text{-}RTP\text{-}list | \geq \sigma_y$ ) then
8                  $Cand = Cand \cup \{C[q]\}$;
9              end
10         end
11     end
12 end
13 return $Cand$
```

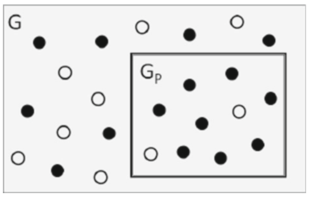


**Fig. 7** Evaluating the predictiveness of $P \Rightarrow y$ compared to more general group $G : G_P \subset G$. The instances with label $y$ are denoted by *filled circle* and the instances without $y$ are denoted by *open circle*

## 5 Mining minimal predictive recent temporal patterns

Although RTP mining focuses the search on temporal patterns that are potentially useful for predicting the labels (e.g., whether or not the adverse medical event would happen), not all frequent RTPs are as important for prediction.

In this section, we describe our approach for selecting the most predictive RTPs. First, we describe the Bayesian score for evaluating the predictiveness of patterns (Sect. 5.1). After that, we argue that scoring each RTP individually and selecting the top scoring ones often leads to spurious patterns (Sect. 5.2). Then, we describe the concept of MPRTP which considers the relations between patterns to filter out spurious patterns (Sect. 5.3). Finally, we present an efficient mining algorithm that integrates MPRTP evaluation with frequent RTP mining and applies efficient pruning techniques to speed up the mining (Sect. 5.4).

### 5.1 The Bayesian score

The Bayesian score we describe here allows us to evaluate how predictive is an RTP $P$ for label $y$ compared to a more general group of instances $G$ (e.g., $G$ might be the entire dataset). We denote this score by $\boldsymbol{BS(P \Rightarrow y, G)}$.

Let $G_P$ denote the group of instances (MSS) in $G$ for which $P$ is an RTP:

$$G_P = \left\{ Z_i : Z_i \in G \wedge R_g(P, Z_i) \right\}$$

Intuitively, the score of $P \Rightarrow y$ compared to $G$ should be high if there is a strong evidence to support the hypothesis that the probability of observing label $y$ in $G_P$ is higher than the probability of observing $y$ in $G$ : $\Pr(y|G_P) > \Pr(y|G)$ (see Fig. 7).

The main idea behind the Bayesian score is to apply Bayesian inference and treat the estimated probabilities as uncertain random variables, which is done by modeling the posterior distribution of the estimated probabilities and integrating over all possible values. We have originally proposed this score in Batal et al. [5] in the context of mining predictive itemset patterns (for atemporal data). The Bayesian score was shown to be more robust compared to classical rule evaluation scores (see [15]), which only rely on point estimation and cannot model the uncertainty of the estimation.

To introduce the Bayesian score, we first define the following three competing models (hypothesis):

1. $M_e$ conjectures that the probability of $y$ is the ***same*** for all instances in $G$.
2. $M_h$ conjectures that the probability of $y$ in $G_P$ is ***higher*** than the probability of $y$ outside $G_P$ ($G \backslash G_P$).
3. $M_l$ conjectures that the probability of $y$ in $G_P$ is ***lower*** than the probability of $y$ outside $G_P$.

We define $BS(P \Rightarrow y, G)$ to be the posterior probability of model $M_h$ (the hypothesis of interest):

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

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

In order to solve Eq. 1, we need to compute the marginal likelihood of all models: $\Pr(G|M_e)$, $\Pr(G|M_h)$ and $\Pr(G|M_l)$. The likelihood of each model given the values of its parameters follows the binomial distribution, while the prior distribution on the parameters is a beta distribution (the beta distribution is a conjugate prior to the binomial distribution, which makes the Bayesian computation tractable). If we do not have prior knowledge about the parameters, we simply use uniform beta distributions.

In the "Appendix", we derive the closed-form solutions for all three marginal likelihoods. We also demonstrate that the complexity of computing the Bayesian score is $O(min(N_{11}, N_{12}, N_{21}, N_{22}))$, where $N_{11}$ is the number of instances in $G_P$ with label $y$, $N_{12}$ is the number of instances in $G_P$ without label $y$, $N_{21}$ is the number of instances in $G \backslash G_P$ (the instances of $G$ outside $G_P$) with label $y$, and $N_{21}$ is the number of instances in $G \backslash G_P$ without label $y$.

## 5.2 Spurious RTPs

A straightforward approach for selecting the most predictive RTPs is to score each RTP by itself and then select the top scoring ones. However, this approach is not effective because it leads to many spurious RTPs, as we explain in the following example.

*Example 7* Assume that having elevated creatinine level is an important indicator of chronic kidney disease (CKD). If we denote this pattern by $P$ =(*Creatinine, High*), we expect the probability of CKD in $G_P$ (the group of instances for which $P$ is an RTP) to be significantly higher than the probability of CKD in the entire population of patients $D$: $\Pr(\text{CKD}|G_P) > \Pr(\text{CKD}|D)$.

Now, consider a pattern $P'$ that is a *b-extension* of $P$, for example, $P'$ : (*Cholesterol, Normal*) *before* (*Creatinine, High*). We know that $G_{P'} \subseteq G_P$ according to Proposition 1. Assume that observing $P'$ does not change our belief about the presence of CKD compared to observing $P$: $\Pr(\text{CKD}|G_{P'}) \approx \Pr(\text{CKD}|G_P)$. In other words, the instances in $G_{P'}$ can be seen as a random subsample of the instances in $G_P$ with respect to the labels of CKD.

The problem is that if we score $P'$ by comparing it to the entire data $D$ (whether using our Bayesian score or using another rule evaluation score [15]), we may conclude that $P'$ is important for predicting CKD because $\Pr(\mathrm{CKD}|G_{P'})$ is significantly higher than $\Pr(\mathrm{CKD}|D)$. However, $P'$ is redundant given the real predictive pattern $P$.

In general, spurious RTPs are formed by extending predictive RTPs with additional irrelevant states. Having spurious RTPs in the result is undesirable for knowledge discovery because they overwhelm the domain expert and prevent him/her from understanding the real patterns in data. It is also undesirable for prediction because they lead to many redundant and highly correlated features, which might hurt the prediction performance. In order to filter out spurious RTPs, we present the concept of *minimal predictive recent temporal patterns*.

## 5.3 Minimal predictive recent temporal patterns

**Definition 9** A temporal pattern $P$ is a **minimal predictive recent temporal pattern** (**MPRTP**) for label $y$ if RTP $P$ predicts $y$ significantly better than all of its suffix subpatterns.

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

where $BS$ is the Bayesian score, $G_S$ is the group of instances for which $S$ is an RTP, and $\delta$ is a user specified significance parameter. Note that, by definition, the empty pattern $\Phi$ is a suffix subpattern of any temporal pattern and $G_\Phi$ is the entire data $D$.

This definition means that if $P$ is an MPRTP for $y$, 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 suffix subpatterns. We call such patterns "minimal predictive" because they are more predictive than all of their simplifications, i.e., they do not contain redundant states. Note that a spurious RTP (such as the one described in Example 7) is not going to be an MPRTP because its effect on the class distribution can be explained by a simpler suffix subpattern that covers a larger population.

## 5.4 The mining algorithm

The algorithm presented in Sect. 4.2 describes how to mine all frequent RTPs for label $y$ using $D_y$ (the MSS in the data that belong to $y$). In order to mine MPRTPs for $y$, the algorithm requires another input: $D_{\neg y}$, the MSS in the data that do not have label $y$ (in order to score the predictiveness of patterns).

The most common way for applying pattern mining in the supervised setting is to use the **two-phase** approach as in [12–14,23,26,45,49]. This approach first generates *all* frequent patterns (the first phase) and then evaluates them in order to select the predictive patterns (the second phase). If we were to apply the two-phase approach to mine MPRTPs for $y$, we would perform the following two phases:

1. Phase I: Mine *all* of the frequent RTPs: $\Omega = \{P_1, \ldots, P_m : \mathit{RTP\text{-}sup}_g(P_i, D_y) \geq \sigma_y\}$.
2. Phase II: For each $P_i \in \Omega$, output rule $P_i \Rightarrow y$ if $P_i$ is an MPRTP for $y$ (i.e., $P_i$ satisfies Definition 9).

In contrast to the two-phase approach, our algorithm *integrates pattern evaluation with frequent RTP mining*. This allows us to apply effective pruning techniques that are not applicable in the two-phase approach. In our context, we say that an RTP is pruned if we do not explore any of its b-extensions. Note that in frequent RTP mining (Sect. 4.2), we prune an RTP if it is not frequent because we know that none of its b-extensions would be frequent. The techniques presented here utilize the *predictiveness* of patterns to further restrict the search space of the algorithm. In the following, we present two pruning techniques: The first one is *lossless* and the second is *lossy*.

#### *5.4.1 Lossless pruning*

The MPRTP definition can help us to prune the search space. The idea is that we can *prune* a frequent RTP $P$, i.e., we do not further extend $P$ backward in time, if we guarantee that none of its backward extension superpatterns is going to be an MPRTP. We know that for any $P'$ that is a *b-extension* of $P$, 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 b-extension* of $P$ with respect to $y$, denoted as $P^*$, to be a hypothetical pattern that is an RTP in *all* and *only* the instances of $G_P$ that have class label $y$:

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

Clearly, $P^*$ is the best possible *b-extension* of $P$ for predicting $y$ (as it covers all instances from $y$ and none of the other instances). Now, we *safely* prune $P$ if $P^*$ does not satisfy the MPRTP definition because then we guarantee that no *b-extension* of $P$ is going to be an MPRTP (hence the technique is lossless).

#### *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 MPRTPs. The idea is that if we are mining MPRTPs for $y$, we prune a frequent RTP $P$ if we have evidence that the underlying probability of $y$ in $G_P$ is lower than the probability of $y$ in the data $D$. To do so, we apply the Bayesian score to evaluate rule $P \Rightarrow y$ compared to $D$ and we prune $P$ if model $M_l$ (the model that assumes the probability of $y$ in $G_P$ is lower that outside $G_P$) is the most likely model. The rationale for this heuristic is that for any $P'$ that is a *b-extension* of $P$, we know that $G_{P'} \subseteq G_P$ (Proposition 1). So if $\Pr(Y = y|G_P)$ is relatively low, we also expect $\Pr(Y = y|G_{P'})$ to be low as well. Thus, $P'$ is unlikely to be an MPRTP for $y$. Note that this heuristic only extends promising RTPs and prunes unpromising ones.

## **6 Experimental evaluation**

This section presents our experimental evaluation. We first describe the two real-world EHR datasets we used in the experiments (Sect. 6.1). Then, we compare RTP and MPRTP with other baselines for the task of detecting adverse medical events (Sect. 6.2). After that, we show the top patterns that MPRTP extracted from the data (Sect. 6.3). Finally, we compare the efficiency of our algorithms with other time-interval pattern mining algorithms (Sect. 6.4).

### 6.1 Temporal datasets

#### *6.1.1 Diabetes dataset*

These data consist of 13,558 electronic health records of adult diabetic patients (both types I and II diabetes). The task is to learn models that can detect various types of disorders that are frequently associated with diabetes.

Each patient's record consists of time series of 19 different laboratory values, including blood glucose, creatinine, glycosylated hemoglobin, blood urea nitrogen, liver function tests, and cholesterol. 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 a medical expert into the following eight major disease categories:

- 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 ten total laboratory measurements or that span less than 3 months (short instances). To make the datasets balanced, we choose the same number of negatives as the number of positives for each category (by randomly subsampling the negatives).

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 state sequences using value abstractions (see Sect. 3.1). 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 temporal abstractions that indicate whether or not the patient has been diagnosed with the disease.

Table 1 summarizes the characteristics of the diabetes datasets, where a different dataset is defined for each of the eight major diseases. 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 abstraction alphabet size ($|\Sigma|$) is 6. Since we have 19 laboratory variables and seven disease variables (the eight major diseases minus the one we are predicting), we have 26 temporal variables per instance. The last column shows the total number of state intervals for each dataset.

### *6.1.2 HIT dataset*

These data are acquired from a database that contains electronic health records of post cardiac surgical patients [17,18]. The task is to learn a model 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

**Table 1** Characteristics of the diabetes datasets

| Dataset | # Instances | # Variables | Alphabet size | # Intervals |
|---|---|---|---|---|
| CARDI | 5,486 | 26 | 6 | 235,990 |
| RENAL | 6,710 | 26 | 6 | 327,957 |
| PERIP | 6,740 | 26 | 6 | 325,872 |
| NEURO | 4,386 | 26 | 6 | 240,572 |
| METAB | 1,936 | 26 | 6 | 118,378 |
| INFLM | 4,788 | 26 | 6 | 264,541 |
| OCULR | 4,490 | 26 | 6 | 227,708 |
| CEREB | 5,648 | 26 | 6 | 319,695 |

thrombosis (blood clot). HIT is a life-threatening condition if it is not detected and managed properly. Hence, it is very important to detect its onset.

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/she is developing HIT. Therefore, an HPF4 test order is a good surrogate for the *HIT-risk* label. Our dataset contains 220 positive instances (*HIT-risk*), and we randomly select 220 negative instances (*no HIT-risk*). The positives include clinical information up to the time HFP4 was first ordered. The negatives were selected from the remaining patients, and they include clinical information up to a randomly selected time point in the patient's record.

For each instance, we consider five 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 (laboratory variables) are numeric time series, so we convert them into time-interval state sequences using both trend abstractions and value abstractions (see Sect. 3.1). Heparin orders (already in an interval-based format) specify the time period during which the patient was taking heparin. For this variable, we simply use temporal abstractions that indicate whether or not the patient is currently on heparin.

## 6.2 Classification performance

In this section, we test the ability of RTP and MPRTP in representing and capturing temporal patterns that are important for the event detection task.

### *6.2.1 Compared methods*

We compare several feature construction methods, which create different feature vector representations of EHR data in order to apply a standard classifier. We did not compare with other time series classification methods such as [7,10,25,38,43,46,51] because these methods are not directly applicable to the multivariate irregularly sample EHR data.[6]

In particular, we compare the performance of the following 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.

[6] The observations of the clinical variables are irregular in time because they are measured asynchronously at different time moments.

2. ***TP:*** The features correspond to all frequent temporal patterns.

$$\left\{P_1, \ldots, P_m : \ sup(P_i, D_y) \geq \sigma_y\right\}$$

   where $sup(P, D_y) = |\{Z_i : Z_i \in D_y \wedge P \in Z_i\}|$ (see Definition 3).
3. ***TP-IG:*** The features correspond to the top **50** frequent temporal patterns, where the patterns are scored using Information Gain (IG) as in [34].
4. ***RTP:*** The features correspond to all frequent RTPs.

$$\left\{P_1, \ldots, P_k : \ RTP\text{-}sup_g(P_i, D_y) \geq \sigma_y\right\}$$

   where $RTP\text{-}sup_g(P, D_y) = |\left\{Z_i : Z_i \in D_y \ \wedge \ R_g(P_j, Z_i)\right\}|$ (see Definition 5).
5. ***RTP-IG:*** The features correspond to the top **50** frequent RTPs, where the RTPs are scored using IG.
6. ***MPRTP:*** The features correspond to the top **50** frequent RTPs, where only the RTPs that satisfy the MPRTP definition (Definition 9 using a significance parameter $\delta = 0.95$) are retained and they are scored according to the Bayesian score (see Sect. 5.1).

The first method (*Last-abs*) is *atemporal* and only considers the most recent abstractions for defining the classification features. On the other hand, methods (2–6) use temporal patterns (built using temporal abstractions and temporal relations) to define features for classification, where a different binary feature is defined for each temporal pattern in the result.

When defining the binary representation of an instance (MSS) $Z_i$ for TP-based methods (*TP* and *TP-IG*), the feature value is set to one if the corresponding temporal pattern is observed anywhere during the instance (Definition 3) and is set to zero otherwise:

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

When defining the binary representation of an instance $Z_i$ for RTP-based methods (*RTP*, *RTP-IG* and *MPRTP*), the feature value is set to one if the corresponding temporal pattern is observed recently in the instance (Definition 5) and is set to zero otherwise:

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

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 different. For example, a temporal pattern may be very discriminative only when observed recently (close to the end of the temporal 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. Moreover, we want to compare using standard univariate scoring (scoring each pattern individually by information gain) with our *MPRTP* approach, which considers the structure of patterns to filter out spurious RTPs.

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 learn a *linear SVM* classifier and evaluate its performance using the area under the ROC curve (*AUC*).

All classification results are reported using averages obtained via *tenfold cross-validation*, where the same train/test splits are used for all compared methods. To evaluate the statistical significance of performance difference, we apply paired *t* tests at 0.05 significance level.[7]

[7] We apply statistical significance testing with *k*-fold cross-validation. In this setting, the testing sets are independent of each other, but the training sets are not. Even though this does not perfectly fit the iid assumption, the significance results are still of great help in comparing different learning methods [27].

**Table 2** *Diabetes dataset*: the area under ROC of the compared feature representation methods for the eight major diabetes diseases (Sect. 6.1.1)

| | *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$*$ | 76.03$*$ | 73.53$*$ | **77.95** | 75.72$*$ | 75.82$*$ |
| NEURO | 72.20$*$ | 74.46$*$ | 72.03$*$ | **76.23** | 71.89$*$ | 74.33$*$ |
| METAB | 80.81$*$ | **83.05** | 80.17$*$ | 81.65$*$ | 80.76$*$ | 82.59 |
| INFLM | 72.21$*$ | 73.20$*$ | 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$*$ |

The best- performing method is shown in bold, and all methods that are statistically inferior to it are marked by $*$. SVM is used for classification

### 6.2.2 *Results on diabetes data*

For all temporal pattern mining methods (*TP*, *TP-IG*, *RTP*, *RTP-IG* and *MPRTP*), we set the local minimum supports ($\sigma_y$) to 15 % of the number of instances in the class.[8] For *RTP*, *RTP-IG* and *MPRTP*, we set the maximum gap parameter (see Definition 5) to 6 months, which was suggested by our medical expert.

Table 2 shows the AUC for each classification task (major disease). We show the best-performing method in boldface, and we show all methods that are statistically significantly inferior to it in gray. Note that features based on temporal patterns (*TP*, *RTP* and *MPRTP*) are beneficial for classification since they outperform features based on only most recent abstractions (*last-abs*). We can see that for most tasks, *RTP* is the best-performing method. Note that although *MPRTP* does not perform as well as *RTP*, it mostly outperforms *RTP-IG* because of its ability to filter out spurious patterns (see, for example, the performance on the *NEURO* dataset in Table 2).

### 6.2.3 *Results on HIT data*

For all temporal pattern mining methods (*TP*, *TP-IG*, *RTP*, *RTP-IG* and *MPRTP*), we set the local minimum supports ($\sigma_y$) to 10 % of the number of instances in the class. For *RTP*, *RTP-IG* and *MPRTP*, we set the maximum gap parameter (see Definition 5) to 2 days, which was suggested by our medical expert. Note that the HIT data are inpatient data and have much finer time granularity than the outpatient diabetes data.

Table 3 shows the AUC on the HIT dataset. We can see that *RTP* and *MPRTP* are the best-performing methods.

## 6.3 Knowledge discovery

In this section, we test the ability of *MPRTP* in finding predictive and non-spurious RTPs that can concisely describe the predicted medical event.

[8] As discussed in Sect. 4.2, we mine frequent patterns for the positives and negatives separately using the local minimum supports.

**Table 3** *HIT dataset*: the area under ROC (AUC) of the compared feature representation methods

| | *Last-abs* | *TP* | *TP-IG* | *RTP* | *RTP-IG* | *MPRTP* |
|---|---|---|---|---|---|---|
| AUC | 87.18$*$ | 90.87 | 87.79$*$ | **91.99** | 88.58$*$ | 91.57 |

The best-performing method is shown in bold, and all methods that are statistically inferior to it are marked by $*$. SVM is used for classification

**Table 4** *Diabetes dataset*: the top MPRTPs with their precision and recall

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

*Dx* diagnosis code (one of the eight major categories described in Sect. 6.1.1), *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

### *6.3.1 Results on diabetes data*

Table 4 shows some of the top MPRTPs according to the Bayesian score on the diabetes data.[9] 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 blood urea nitrogen (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 incorporating prior clinical knowledge, are in accordance with the medical diagnosis guidelines.

### *6.3.2 Results on HIT data*

Table 5 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 [44]. $P_5$ relates the risk of HIT with an increasing trend of APTT. This relation is not obvious from the HIT detection guidelines. However, it has been recently discussed in the literature [37]. Hence, this pattern requires further investigation.

## 6.4 Mining efficiency

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

[9] 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 three predictive MPRTPs for RENAL and the top two MPRTPs for other categories.

**Table 5** *HIT dataset*: the top five MPRTPs with their precision and recall

| *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 |

*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): $< 76 \times 10^9$ per liter; *PLT=L* (Low): $< 118 \times 10^9$ per liter

### *6.4.1 Compared methods*

We compare the running time of the following methods:

1. ***TP_Apriori:*** Mine frequent temporal patterns by extending the Apriori algorithm [2,3] to the time-interval domain.
2. ***TP_lists:*** Mine frequent temporal patterns by applying the vertical data format [52,53] to the time-interval domain as in [8,9].
3. ***RTP_no-lists:*** Mine frequent RTPs as described in Sect. 4.2, but without applying the id-list indexing in Sect. 4.2.3 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:*** Our method for mining frequent RTPs as described in Sect. 4.2 which also applies the id-list indexing.
5. ***MPRTP:*** Our method for mining MPRTPs which applies all optimizations used by the RTP method and also the pruning techniques described in Sects. 5.4.1 and 5.4.2.

To make the comparison fair, *all methods apply the techniques we propose in Sect. 4.2.2 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 3 GHz CPU and 16 GB of RAM.

### *6.4.2 Results on diabetes data*

Figure 8 shows the execution time (on logarithmic scale) of the compared methods on all major diagnosis datasets. Similar to the previous settings (Sect. 6.2.2), we set the local minimum supports to 15 % and the maximum gap parameter to 6 months (unless stated otherwise).

We can see that *RTP* and *MPRTP* are much more efficient than the other temporal pattern mining methods. For example, on the *INFLM* dataset, *RTP* is around five times faster than *TP_lists*, ten times faster than *RTP_no-lists* and 30 times faster than *TP_Apriori*. Furthermore, *MPRTP* is more efficient than *RTP* for all datasets.

Figure 9 compares the execution time of the different methods on the *CARDI* dataset for different minimum support thresholds.

Finally, we examine the effect of the maximum gap parameter ($g$) on the efficiency of recent temporal pattern mining methods (*RTP_no-lists*, *RTP* and *MPRTP*). Figure 10 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 $g$).

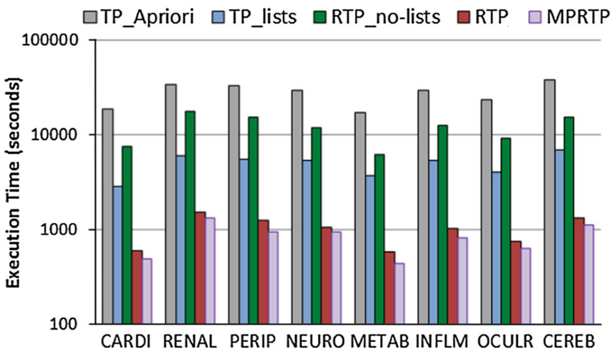


**Fig. 8** *Diabetes dataset*: the mining time (in seconds) of the compared temporal pattern mining methods (Sect. 6.4.1) for the eight major diabetes diseases. The local minimum support is 15 %

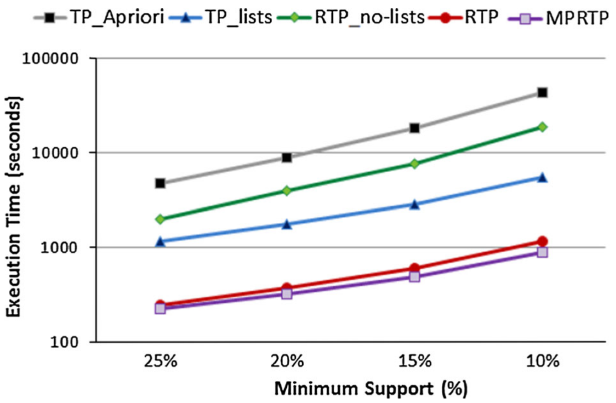


**Fig. 9** *Diabetes dataset (CARDI)*: the mining time (in seconds) of the compared temporal pattern mining methods (Sect. 6.4.1) on the CARDI diabetes dataset for different values of the minimum support

Clearly, the execution time of *RTP_no-lists*, *RTP* and *MPRTP* increases 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 (pruning a candidate *k-pattern* if it contains an infrequent ($k-1$)-*subpattern*) becomes more efficient (generates less candidates) than the backward extension of temporal patterns (see Example 5). On the other hand, the execution time of *RTP* and *MPRTP* increases much slower with $g$ and they maintain their efficiency advantage over *TP_Apriori* and *TP_lists* for larger values of $g$.

### *6.4.3 Results on HIT data*

Figure 11 shows the execution time (on logarithmic scale) of the compared methods. Similar to the previous settings for the HIT data (Sect. 6.2.3), we set the local minimum supports to 10 % and the maximum gap parameter to 2 days. Again, we see that *RTP* and *MPRTP* are more efficient than the other temporal pattern mining methods.

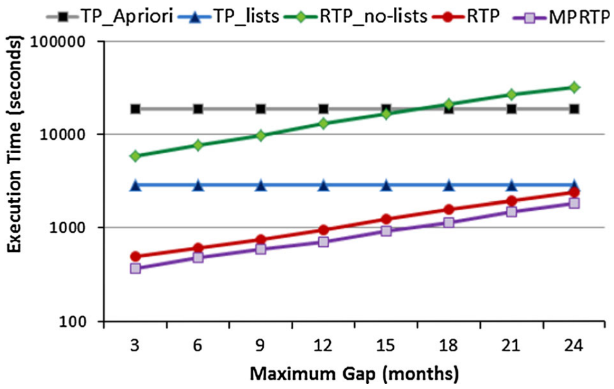


**Fig. 10** *Diabetes dataset (CARDI)*: the mining time (in seconds) of the compared temporal pattern mining methods (Sect. 6.4.1) on the CARDI diabetes dataset for different values of the maximum gap parameter

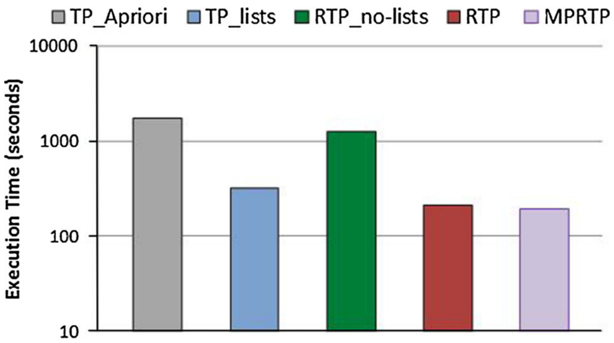


**Fig. 11** *HIT dataset*: the mining time (in seconds) of the compared temporal pattern mining methods (Sect. 6.4.1). The local minimum support is 10 %

## 7 Conclusion

The increasing availability of large temporal data prompts the development of scalable and more efficient mining methods. Methods for mining sequential patterns (time-point patterns) were first introduced in the literature [3,35,50,53]. Later on, these methods have been extended to mine data for which the events have time durations (time-interval patterns) [20,22,28,30,32,39,47,48]. However, mining the entire set of frequent patterns (whether sequential patterns or time-interval patterns) from large-scale data is inherently a computationally expensive task. To alleviate this problem, previous research [36,41] introduced several temporal constraints to scale up the mining, such as restricting the total pattern duration or restricting the permitted gap between consecutive states in a pattern. This paper 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 methods on two clinical event detection tasks. The results showed the following benefits of our methods:

1. RTP and MPRTP are able to learn accurate event detection classifiers for real-world clinical tasks.
2. MPRTP is effective for selecting predictive and non-spurious RTPs, which makes it a useful tool for knowledge discovery.
3. Mining RTPs or MPRTPs is much more scalable than mining all frequent time-interval patterns.

## Appendix: The Bayesian score: mathematical derivation and computational complexity

In Sect. 5.1, we briefly introduced the Bayesian score of RTP $P$ for predicting class label $y$ compared to a more general group $G$: $G_P \subset G$. In this "Appendix", we derive the marginal likelihood for models $M_e$, $M_h$ and $M_l$, which are required for computing the Bayesian score (solving Eq. 1). "The closed-form solution of the marginal likelihood for model $M_e$" of appendix describes the closed-form solution for computing $P(G|M_e)$, which is the marginal likelihood for model $M_e$ (the probability of $y$ is the same inside and outside $G_P$). "Deriving a closed-form solution of the marginal likelihood for model $M_h$" of appendix derives the closed form solution for computing $P(G|M_h)$, which is the marginal likelihood for model $M_h$ (the probability of $y$ in $G_P$ is higher than outside $G_P$). "Four equivalent solutions of the marginal likelihood for model $M_h$" of appendix shows the four equivalent formulas for computing $P(G|M_h)$. "Deriving a closed-form solution of the marginal likelihood for model $M_l$" of appendix illustrates how to obtain the marginal likelihood for model $M_l$ (the probability of $y$ in $G_P$ is lower than outside $G_P$) directly from the solution to $P(G|M_h)$. Finally, "Computational complexity" of appendix analyzes the overall computational complexity for computing the Bayesian score.

### The closed-form solution of the marginal likelihood for model $M_e$

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 having label $Y = y$. Let us denote this probability by $\theta$. To represent our uncertainty about $\theta$, we use a beta distribution with parameters $\alpha$ and $\beta$. Let $N_{*1}$ be the number of instances in $G$ with $Y = y$ and let $N_{*2}$ be the number of instances in $G$ with $Y \neq y$ (i.e., instances that do not have class label $y$). 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 [19]:

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

where $\Gamma$ is the gamma function.

### Deriving a closed-form solution of the marginal likelihood for model $M_h$

Now let us now define the marginal likelihood for model $M_h$. This model assumes that the probability of $Y = y$ for the instances in $G_P$, denoted by $\theta_1$, is higher than the probability of $Y = y$ for the instances of $G$ that are outside $G_P$ ($G \backslash G_P$), denoted by $\theta_2$. To represent our uncertainty about $\theta_1$, we use a beta distribution with parameters $\alpha_1$ and $\beta_1$. To represent

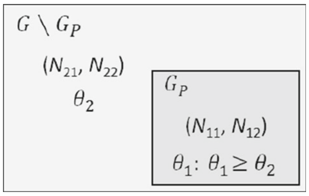


**Fig. 12** A diagram illustrating model $M_h$

our uncertainty about $\theta_2$, we use a beta distribution with parameters $\alpha_2$ and $\beta_2$. 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 outside $G_P$ with $Y = y$ and with $Y \neq y$, respectively (see Fig. 12).

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

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

where $k$ is a normalization constant for the parameter prior. Note that we do not assume that the parameters are independent, but rather we constrain $\theta_1$ to be higher than $\theta_2$.

To solve Eq. 3, we first show how to solve the integral over $\theta_2$ in closed form, which is denoted by $f_2$ in Eq. 3. We then expand the function denoted by $f_1$, multiply it by the solution to $f_2$, and solve the integral over $\theta_1$ in closed form to complete the integration.

We use the regularized incomplete beta function [1] to solve the integral given by $f_2$, which is as follows:

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

where $a$ and $b$ should be natural numbers.

Note that when $x = 1$ in Eq. 4, the solution to the integral in that equation is simply the following:

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

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

$$\begin{aligned}
f_2 &= \int_{\theta_2=0}^{\theta_1} {\theta_2}^{N_{21}} \cdot (1-\theta_2)^{N_{22}} \cdot beta(\theta_2;\alpha_2,\beta_2)d\theta_2 \\
&= \int_{\theta_2=0}^{\theta_1} {\theta_2}^{N_{21}} \cdot (1-\theta_2)^{N_{22}} \cdot \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2)\cdot\Gamma(\beta_2)} \cdot \theta_2^{\alpha_2-1} \cdot (1-\theta_2)^{\beta_2-1} d\theta_2
\end{aligned}$$

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

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

Using Eq. 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:

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

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

Now, we combine Eqs. 6 and 7 to solve Eq. 3:

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

which by Eq. 5 is

$$
\begin{aligned}
Pr(G|M_h) &= \frac{1}{k} \cdot \frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\cdot\Gamma(\beta_1)} \cdot \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2)\cdot\Gamma(\beta_2)} \cdot \frac{\Gamma(a)\cdot\Gamma(b)}{\Gamma(a+b)} \\
&\quad \cdot \sum_{j=a}^{a+b-1} \frac{\Gamma(a+b)}{\Gamma(j+1)\cdot\Gamma(a+b-j)} \cdot \frac{\Gamma(c+j)\cdot\Gamma(a+b+d-1-j)}{\Gamma(a+b+c+d-1)} \\
&= \frac{1}{k} \cdot \frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\cdot\Gamma(\beta_1)} \cdot \frac{\Gamma(\alpha_2+\beta_2)}{\Gamma(\alpha_2)\cdot\Gamma(\beta_2)} \\
&\quad \cdot \sum_{j=a}^{a+b-1} \frac{\Gamma(a)\cdot\Gamma(b)}{\Gamma(j+1)\cdot\Gamma(a+b-j)} \cdot \frac{\Gamma(c+j)\cdot\Gamma(a+b+d-1-j)}{\Gamma(a+b+c+d-1)}
\end{aligned}
\tag{8}
$$

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 Eq. 3 (without the $k$ term) with $N_{11} = N_{12} = N_{21} = N_{22} = 0$. Doing so is equivalent to applying Eq. 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 $\alpha_1 = \beta_1 = \alpha_2 = \beta_2 = 1$.

## Four equivalent solutions of the marginal likelihood for model $M_h$

In the previous section, we showed the full derivation of the closed-form solution of the marginal likelihood for model $M_h$. It turned out that there are four equivalent solutions to Eq. 3. 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 $M_h$ (Eq. 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.

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

$$
\begin{aligned}
C \cdot \Bigg( & \frac{\Gamma(a)\cdot\Gamma(b)}{\Gamma(a+b)} \cdot \frac{\Gamma(c)\cdot\Gamma(d)}{\Gamma(c+d)} - \sum_{j=b}^{a+b-1} \frac{\Gamma(a)\cdot\Gamma(b)}{\Gamma(j+1)\cdot\Gamma(a+b-j)} \\
& \cdot \frac{\Gamma(d+j)\cdot\Gamma(a+b+c-j-1)}{\Gamma(a+b+c+d-1)} \Bigg)
\end{aligned}
\tag{12}
$$

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

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

Deriving a closed-form solution of the marginal likelihood for model $M_l$

Lastly, let us define the marginal likelihood for model $M_l$, which assumes that the probability of $Y=y$ for the instances in $G_P$ ($\theta_1$) is lower than the probability of $Y=y$ for the instances of $G$ that are outside $G_P$ ($\theta_2$). The marginal likelihood for $M_l$ is similar to Eq. 3, but integrates $\theta_2$ from 0 to 1 and constrains $\theta_1$ to be integrated from 0 to $\theta_2$ (forcing $\theta_1$ to be smaller than $\theta_2$) as follows:

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

By solving the integral given by $f_2$, we get:

$$\begin{aligned}f_2&=\frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\cdot\Gamma(\beta_1)}\int_{\theta_1=0}^{\theta_2}{\theta_1}^{c-1}\cdot(1-\theta_1)^{d-1}d\theta_2\\&=\frac{\Gamma(\alpha_1+\beta_1)}{\Gamma(\alpha_1)\cdot\Gamma(\beta_1)}\cdot\frac{\Gamma(c)\cdot\Gamma(d)}{\Gamma(c+d)}\cdot\sum_{j=c}^{c+d-1}\frac{\Gamma(c+d)}{\Gamma(j+1)\cdot\Gamma(c+d-j)}\cdot\theta_2^j\cdot(1-\theta_2)^{c+d-1-j}\end{aligned}$$

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

By solving $f_1$, we get:

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

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

Now, we can solve Eq. 14:

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

where $C$ is the same constant we defined by Eq. 9 in the previous section.

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

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

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

Computational complexity

Since we require that $N_{11}, N_{12}, N_{21}, N_{22}, \alpha_1, \beta_1, \alpha_2$ and $\beta_2$ 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 Eqs. 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, Eqs. 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 Eqs. 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 Eq. 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 Eq. 2. $Pr(M_h|G)$ can be computed by applying Eqs. 10, 11, 12 or 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 Eq. 16. By assuming that $\alpha_1, \beta_1, \alpha_2, \beta_2$ bounded from above, the overall complexity for computing the Bayesian score is $\boldsymbol{O(min(N_{11}, N_{12}, N_{21}, N_{22}))}$.

## References

1. Abramowitz M, Stegun IA (1964) Handbook of mathematical functions with formulas, graphs, and mathematical tables
2. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Proceedings of the international conference on very large data bases (VLDB)
3. Agrawal R, Srikant R (1995) Mining sequential patterns. In: Proceedings of the international conference on data engineering (ICDE)
4. Allen F (1984) Towards a general theory of action and time. Artif Intell 23:123–154
5. Batal I, Cooper G, Hauskrecht M (2012) 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)
6. Batal I, Fradkin D, Harrison J, Moerchen F, Hauskrecht M (2012) 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)
7. Batal I, Hauskrecht M (2009) A supervised time series feature extraction technique using DCT and DWT. In: International conference on machine learning and applications (ICMLA)

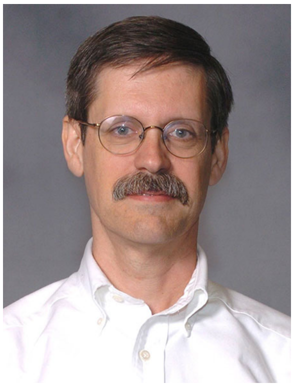
8. Batal I, Valizadegan H, Cooper GF, Hauskrecht M (2011) A pattern mining approach for classifying multivariate temporal data. In: Proceedings of the IEEE international conference on bioinformatics and biomedicine (BIBM)
9. Batal I, Valizadegan H, Cooper GF, Hauskrecht M (2013) A temporal pattern mining approach for classifying electronic health record data. ACM Trans Intell Syst Technol 4(4). doi:10.1145/2508037.2508044
10. Blasiak S, Rangwala H (2011) A hidden Markov model variant for sequence classification. In: Proceedings of the international joint conferences on artificial intelligence (IJCAI)
11. Chandola V, Eilertson E, Ertoz L, Simon G, Kumar V (2006) Data mining for cyber security. In: Data warehousing and data mining techniques for computer security. Springer, Berlin
12. Cheng H, Yan X, Han J, wei Hsu C (2007) Discriminative frequent pattern analysis for effective classification. In: Proceedings of the international conference on data engineering (ICDE)
13. Deshpande M, Kuramochi M, Wale N, Karypis G (2005) Frequent substructure-based approaches for classifying chemical compounds. IEEE Trans Knowl Data Eng 17:1036–1050
14. Exarchos TP, Tsipouras MG, Papaloukas C, Fotiadis DI (2008) A two-stage methodology for sequence classification based on sequential pattern mining and optimization. Data Knowl Eng 66:467–487
15. Geng L, Hamilton HJ (2006) Interestingness measures for data mining: a survey. ACM Comput Surv 38(3)
16. Guttormsson SE, Marks RJ, El-Sharkawi MA, Kerszenbaum I (1999) Elliptical novelty grouping for on-line short-turn detection of excited running rotors. IEEE Trans Energy Convers 14(1):16–22
17. Hauskrecht M, Batal I, Valko M, Visweswaram S, Cooper G, Clermont G (2012) Outlier detection for patient monitoring and alerting. J Biomed Inform 46(1):47–55
18. Hauskrecht M, Valko M, Batal I, Clermont G, Visweswaram S, Cooper G (2010) Conditional outlier detection for clinical alerting. In Proceedings of the American Medical Informatics Association (AMIA)
19. Heckerman D, Geiger D, Chickering DM (1995) Learning Bayesian networks: the combination of knowledge and statistical data. Mach Learn 20:197–243
20. Höppner F (2001) Discovery of temporal patterns. Learning rules about the qualitative behaviour of time series. In: Proceedings of the European conference on principles of data mining and knowledge discovery (PKDD)
21. Höppner F (2003) Knowledge discovery from sequential data, PhD thesis. Technical University Braunschweig, Germany
22. Kam P-S, Fu AW-C (2000) Discovering temporal patterns for interval-based events. In: Proceedings of the international conference on data warehousing and knowledge discovery (DaWaK)
23. Kavsek B, Lavrač N (2006) APRIORI-SD: adapting association rule learning to subgroup discovery. Appl Artif Intell 20(7):543–583
24. Keogh E, Chu S, Hart D, Pazzani M (1993) Segmenting time series: a survey and novel approach. In: Data mining in time series databases. World Scientific, pp 1–22
25. Li L, Prakash BA, Faloutsos C (2010) Parsimonious linear fingerprinting for time series. PVLDB 3:385–396
26. Li W, Han J, Pei J (2001) CMAR: accurate and efficient classification based on multiple class-association rules. In: Proceedings of the international conference on data mining (ICDM)
27. Mitchell TM (1997) Machine learning. McGraw-Hill Inc., New York
28. Moerchen F (2006a) Algorithms for time series knowledge mining. In: Proceedings of the international conference on knowledge discovery and data mining (SIGKDD)
29. Moerchen F (2006b) Time series knowledge mining, PhD thesis. Philipps-University Marburg
30. Moskovitch R, Shahar Y (2009), Medical temporal-knowledge discovery via temporal abstraction. In: Proceedings of the American Medical Informatics Association (AMIA)
31. Papadimitriou S, Sun J, Faloutsos C (2005) Streaming pattern discovery in multiple time-series. In: Proceedings of the international conference on very large data bases (VLDB)
32. Papapetrou P, Kollios G, Sclaroff S, Gunopulos D (2005) Discovering frequent arrangements of temporal intervals. In Proceedings of the international conference on data mining (ICDM)
33. Patel D, Hsu W, Lee ML (2008a) Mining relationships among interval-based events for classification. In: Proceedings of the international conference on management of data (SIGMOD)
34. Patel D, Hsu W, Lee ML (2008b) Mining relationships among interval-based events for classification, In: Proceedings of the international conference on management of data (SIGMOD)
35. Pei J, Han J, Mortazavi-asl B, Pinto H, Chen Q, Dayal U, Hsu MC (2001) PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of the international conference on data engineering (ICDE)
36. Pei J, Han J, Wang W (2007) Constraint-based sequential pattern mining: the pattern-growth methods. J Intell Inf Syst 28:133–160

37. Pendelton R, Wheeler M, Rodgers G (2006) Argatroban dosing of patients with heparin induced thrombocytopenia and an elevated aPTT due to antiphospholipid antibody syndrome. Ann Pharmacother 40:972–976
38. Ratanamahatana C, Keogh EJ (2005) Three myths about dynamic time warping data mining, In: Proceedings of the SIAM international conference on data mining (SDM)
39. Sacchi L, Larizza C, Combi C, Bellazzi R (2007) Data mining with temporal abstractions: learning rules from time series. Data Min Knowl Discov 15(2):217–247
40. Shahar Y (1997) A framework for knowledge-based temporal abstraction. Artif Intell 90:79–133
41. Srikant R, Agrawal R (1996) Mining sequential patterns: generalizations and performance improvements. In: Proceedings of the international conference on extending database technology (EDBT)
42. Srivastava A, Kundu A, Sural S, Majumdar AK (2008) Credit card fraud detection using hidden Markov model. IEEE Trans Dependable Secure Comput 5(1):37–48
43. Vail DL, Veloso MM, Lafferty JD (2007) Conditional random fields for activity recognition. In: Proceedings of the international joint conference on autonomous agents and multiagent systems (AAMAS)
44. Warkentin T (2000) Heparin-induced thrombocytopenia: pathogenesis and management. Br J Haematol 121:535–555
45. Webb GI (2007) Discovering significant patterns. Mach Learn 68(1):1–33
46. Weng X, Shen J (2008) Classification of multivariate time series using two-dimensional singular value decomposition. Knowl Based Syst 21(7):535–539
47. Winarko E, Roddick JF (2007) ARMADA—an algorithm for discovering richer relative temporal association rules from interval-based data. Data Knowl Eng 63:76–90
48. Wu S-Y, Chen Y-L (2007) Mining nonambiguous temporal patterns for interval-based events. IEEE Trans Knowl Data Eng 19:742–758
49. Xin D, Cheng H, Yan X, Han J (2006) Extracting redundancy-aware top-k patterns. In: Proceedings of the international conference on knowledge discovery and data mining (SIGKDD)
50. Yan X, Han J, Afshar R (2003) CloSpan: mining closed sequential patterns in large datasets. In: Proceedings of the SIAM international conference on data mining (SDM)
51. Yang K, Shahabi C (2004) A PCA-based similarity measure for multivariate time series. In: Proceedings of the international workshop on multimedia databases
52. Zaki MJ (2000) Scalable algorithms for association mining. IEEE Trans Knowl Data Eng 12:372–390
53. Zaki MJ (2001) SPADE: an efficient algorithm for mining frequent sequences. Mach Learn 42:31–60

**Iyad Batal** obtained his Ph.D. in Computer Science from the University of Pittsburgh. His thesis contributed novel and efficient methods for subgroup discovery, temporal pattern mining, and their applications to electronic medical records. He has also worked on several other research topics, including multi-dimensional time series analysis, multi-label classification and probabilistic graphical models. He is currently a Research Scientist at GE Global Research, working on designing machine learning solutions for solving real-world industrial problems and modeling sensor time series data.

**Gregory F. Cooper** M.D., Ph.D. is Professor of Biomedical Informatics and of Intelligent Systems at the University of Pittsburgh. His research involves the use of decision theory, probability theory, Bayesian statistics, machine learning, and artificial intelligence to address biomedical informatics research problems. Current research includes the development, implementation, and evaluation of computer-based methods for clinical alerting based on machine learning, causal discovery from biomedical data, patient outcome prediction, and disease outbreak detection in the population based on data in electronic medical records.

**Dmitriy Fradkin** is a Senior Scientist at Siemens Corporate Technology, Princeton NJ. He received B.A. in Mathematics and Computer Science from Brandeis University, Waltham, MA in 1999 and Ph.D. from Rutgers, The State University of New Jersey in 2006. Before joining Siemens in 2007 he has worked at Ask.com. His research is in applying data mining and machine learning techniques to solve real-world problems is areas of predictive maintenance, healthcare and text analytics. Dr. Fradkin is a member of the ACM SIGKDD, and a reviewer for several data mining journals



**James Harrison Jr.** M.D., Ph.D. is Associate Professor of Public Health Sciences and Pathology, and Director of the Division of Biomedical Informatics in the Department of Public Health Sciences at the University of Virginia. He has 20 years of experience in the field of medical informatics, including work in clinical laboratory information systems, electronic health records, clinical data analysis, and informatics research grant support from the National Library of Medicine and the National Cancer Institute. He leads clinical research informatics in the UVA School of Medicine with primary responsibility for the institutional Clinical Data Repository, the Cancer Clinical Trials Management System, the Biorepository Information System, and a locally-developed general purpose framework for research databases.

**Fabian Moerchen** graduated with a Ph.D. in 2006 from the Philipps University of Marburg, Germany with summa cum laude. His thesis contributed novel methods in mining temporal pattern from interval data. From 2006–2012 worked at Siemens Corporate Research leading data mining projects with applications in predictive maintenance, text mining, healthcare, and sustainable energy. He continued his research in temporal data mining in the context of industrial and scientific problems and served the community as a reviewer, organizer of workshops, and presenter of tutorials. Since 2012 he is leading a data science team at Amazon to improve customer experience using machine learning and big data analytics.

**Milos Hauskrecht** is an Associate Professor of Computer Science at the University of Pittsburgh. He received his Ph.D. from MIT in 1997. His research interest are in probabilistic modeling, machine learning, data mining and their applications in medicine. He has authored or co-authored over 100 publications in these areas. His research is funded by grants from NIH and NSF. He serves regularly on program committees of top artificial intelligence and biomedical informatics conferences. He is the recipient of Homer R. Warner award for 2010 for his work on outlier-based clinical monitoring and alerting.