



HAL
open science

Mining temporal patterns with quantitative intervals

Thomas Guyet, René Quiniou

► **To cite this version:**

Thomas Guyet, René Quiniou. Mining temporal patterns with quantitative intervals. 4th International Workshop on Mining Complex Data, (IEEE ICDM) Workshop, 2008, Italy. pp.10. hal-00431445

HAL Id: hal-00431445

<https://hal.science/hal-00431445>

Submitted on 13 Nov 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Mining temporal patterns with quantitative intervals

Thomas Guyet
René Quiniou
INRIA, DREAM Team
Campus de Beaulieu, 35042 Rennes, France
firstname.lastname@irisa.fr

Abstract

In this paper we consider the problem of discovering frequent temporal patterns in a database of temporal sequences, where a temporal sequence is a set of items with associated dates and durations. Since the quantitative temporal information appears to be fundamental in many contexts, it is taken into account in the mining processes and returned as part of the extracted knowledge. To this end, we have adapted the classical APriori [1] framework to propose an efficient algorithm based on a hyper-cube representation of temporal sequences. The extraction of quantitative temporal information is performed using a density estimation of the distribution of event intervals from the temporal sequences. An evaluation on synthetic data sets shows that the proposed algorithm can robustly extract frequent temporal patterns with quantitative temporal extents.

1. Introduction

The classical view of frequent sequential patterns mining (FSPM) [1] considers a large database of sequences \mathcal{D} . Each sequence $s \in \mathcal{D}$ is an ordered set of items with possible duplications. The mining task consists in extracting the longest patterns that are sufficiently frequent, *i.e.* that can be found in a number of sequence of \mathcal{D} above a predefined threshold.

The FSPM is strongly motivated by its application to knowledge extraction from customer purchase database [1], DNA sequences, web logs [15], medical time series [9, 16], etc. Most of these investigations have only considered a linear sequential notion of time. But, in some contexts, the quantitative temporal dimension of data provides fundamental information that should be taken into account in the mining processes and returned as part of the extracted knowledge. For example, in web log analysis, the browsing patterns can be refined by the time spent to explore the web pages. The comparison between the quantitative time spent on a web page and the required time to browse its content

may be useful to decide whether some part of the information has not been read. In a medical context, the temporal information is crucial to diagnose the patients accurately. This is obvious in situations ranging from diabetes mellitus to intensive care unit ECG monitoring and artificial ventilation. For example, in [16] different kinds of cardiac arrhythmias could be discriminated thanks to the quantitative temporal intervals between events (chronicles models).

Some authors, *e.g.* Chiu et al. [6], propose to simulate event duration by the repetition of a same event having a fixed atomic duration. However, in some applications it is necessary to distinguish the repetition of events from a single occurrence of this event: for example, two apneas of 30s each should not be confused with a single apnea of 1min. This distinction is only possible with a quantitative representation of the time information.

Unfortunately, discovering the quantitative temporal extents of events is a difficult problem. The construction of the sequence of items should be intertwined with the construction of representative durations. On the one hand, temporal sequence instances are required to construct the representative duration between two events in a temporal sequence. But, on the other hand, the durations are required to identify the temporal sequence instances. To tackle this problem, we propose to enhance the classical *APriori* algorithm to handle quantitative temporal extents of the items.

In section 2, we define the temporal sequence, the hyper-cube representation and temporal pattern. Then in section 3, we give some details about temporal pattern extraction task. In section 4, the algorithms are presented and the section 5 presents the evaluations. Finally, we present related work in section 6.

2. Temporal sequences and temporal patterns

2.1. Temporal sequences

Definition 1. (Temporal sequence) *A temporal sequence \mathcal{S} is a set of temporal items, where a temporal item \mathcal{A} , denoted as $\mathcal{A} = (A, [l, u])$, is a symbolic event A associated*

with a non empty interval $[l, u]$, where $l, u \in \mathbb{R}, l < u$, are timestamps:

$$\mathcal{S} = \{(s_i, [l_i, u_i])\}_{i \in \mathbb{N}_n}$$

n , the number of temporal items, is called the dimension of the sequence \mathcal{S} .

Notice that the delay between two events is indirectly given by the beginning instants of the event intervals, and that there is no assumption on their occurrence. In particular, event-overlapping is possible.

The temporal sequences may represent several runs of a dynamic process that generates these events, or may be extracted from a sliding window over long temporal sequences. In the latter case, the timestamps are translated in order to represent the relative time *within* the window (and not the absolute time). So, the temporal sequences can be compared with each other.

Definition 2. (Temporal sub-sequence) A temporal sub-sequence of $\mathcal{S} = \{(s_i, [l_i, u_i])\}_{i \in \mathbb{N}_n}$, is a temporal sequence \mathcal{S}' of dimension $m < n$, such that $\mathcal{S}' = \{(s_j, [l_j, u_j])\}_{j \in P(\mathbb{N}_n)}$, where $P(\mathbb{N}_n)$ is a subset of \mathbb{N}_n .

Definition 3. (Symbolic signature) The symbolic signature of a temporal sequence $\mathcal{S} = \{(s_i, [l_i, u_i])\}_{i \in \mathbb{N}_n}$ is the ordered sequence of the symbolic events : $\underline{\mathcal{S}} = \{s_i \mid l_i < l_{i+1}\}_{i \in \mathbb{P}_n}$, where \mathbb{P}_n is a permutation of \mathbb{N}_n .

Example 1. (Temporal sequence) Let $s = \{(A, [1.1, 2.9]), (B, [4, 7.8]), (C, [2, 5.3])\}$ be a temporal sequence. s can be graphically represented as in Figure 1. The example indicates that event A occurs at the time unit 1.1 and spans 1.8 time units. Then, B occurs at time unit 4 and spans 3.8 time units. In parallel, event C occurs at the time unit 2 and spans 3.3 time units. C overlaps A and B.

The symbolic signature of the s is $\underline{s} = \{A, C, B\}$.

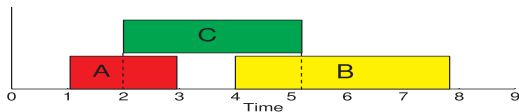


Figure 1. Representation of the temporal sequence s from example 1.

Now, we introduce the hyper-cube representation of a temporal sequence. It will be used to formalize the pattern extraction principle.

Definition 4. (Hyper-cube) An hyper-cube H of dimension n is a geometrical volume in \mathbb{R}^n . It is defined by the bounds $\{l_i, u_i\}_{i \in \mathbb{N}}$:

$$\forall x \in \mathbb{R}^n, x \in H \Leftrightarrow \forall i \in \mathbb{N}, l_i \leq x_i \leq u_i.$$

Definition 5. (Hyper-cube representation of a temporal sequences) Given a temporal sequence \mathcal{S} of dimension n , the temporal bounds of \mathcal{S} define a unique hyper-cube in \mathbb{R}^n . The temporal bounds of the sequence items are determined by the orthogonal projection along each corresponding timeline (axis of \mathbb{R}^n).

If some symbolic event has several occurrences in a temporal sequence, all the related dimensions are distinguished.

Example 2. (Hyper-cube representation of a temporal sequence) Given the temporal sequence of example 1, Figure 2 shows the hyper-cube representation of s . The dimension of s is 3, then the representation belongs to \mathbb{R}^3 . The first, second and third dimensions are the timelines that determine the bounds of A, B and C respectively.

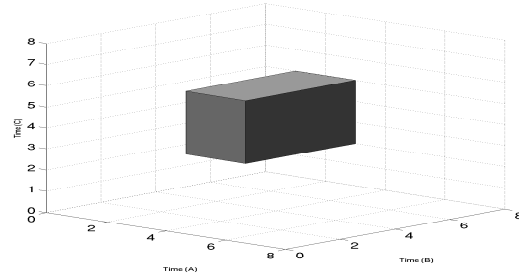


Figure 2. Hyper-cube representation of the temporal sequence (dimension 3).

This representation of temporal sequences is used to define the similarity between two temporal sequences \mathcal{S}^1 and \mathcal{S}^2 as the normalized overlapping volume of their hyper-cube representations, i.e. the volume of the intersection of the representations of \mathcal{S}^1 and \mathcal{S}^2 divided by the volume of the biggest hyper-cube.

Definition 6. (Temporal sequences similarity) Let \mathcal{S}^1 and \mathcal{S}^2 be two temporal sequences, then the similarity between \mathcal{S}^1 and \mathcal{S}^2 , denoted by $\text{sim}(\mathcal{S}^1, \mathcal{S}^2)$, is defined as:

$$\text{sim}(\mathcal{S}^1, \mathcal{S}^2) = \begin{cases} 0 & \text{if } \underline{\mathcal{S}}^1 \neq \underline{\mathcal{S}}^2 \\ \frac{\prod_{i=1}^n \max(0, \min(u_i^1, u_i^2) - \max(l_i^1, l_i^2))}{\max(\prod_{i=1}^n u_i^1 - l_i^1, \prod_{i=1}^n u_i^2 - l_i^2)} & \text{if } \underline{\mathcal{S}}^1 = \underline{\mathcal{S}}^2 \end{cases}$$

where $\underline{\mathcal{S}}^1 = \underline{\mathcal{S}}^2 \Leftrightarrow \forall i \in \mathbb{N}_n, \underline{s}_i^1 = \underline{s}_i^2$

$\max(0, \min(u_i^1, u_i^2) - \max(l_i^1, l_i^2))$ gives the size of the intersection of two intervals. Then, the similarity is the product of these sizes over the maximum product of the interval sizes of \mathcal{S}^1 and \mathcal{S}^2 , i.e. the normalized volume of the intersection. This function is not a metric, but assuming that intervals cannot be empty, it is easy to prove that the proposed similarity is a positive-definite and symmetric function.

2.2. Temporal patterns

Definition 7. (Temporal pattern) A temporal pattern is a temporal sequence ($\mathcal{P} = \{(p_i, [l_i, u_i])\}_{i \in \mathbb{N}_n}$) that is the representative of a set of similar temporal sequences.

Similarly to Giannotti et al. [9], a pattern is a representative. However, in [9], patterns are rules $a \stackrel{t}{\prec} b$, meaning that the representative duration between events a and b is t . Our patterns provides the delay between events as well as the duration of events. Chronicles [8] or delta patterns [17] are not representatives. They just generalize the represented temporal instances. In fact, the interval between two events gives the lower bound and the upper bound of the delay between these two events.

Definition 8. (Temporal pattern coverage) A temporal pattern $\mathcal{P} = \{(p_i, [l_i^p, u_i^p])\}_{i \in \mathbb{N}_n}$ covers an example $\mathcal{S} = \{(s_i, [l_i^s, u_i^s])\}_{i \in \mathbb{N}_m}$, denoted $\mathcal{S} \preceq \mathcal{P}$, if the symbolic signature of \mathcal{S} contains the whole symbolic signature of \mathcal{P} , i.e.:

$$\exists f : \mathbb{N}_n \mapsto \mathbb{N}_m \text{ injective, } \forall i \in \mathbb{N}_n, p_i = s_{f(i)}$$

If $\mathcal{S} \preceq \mathcal{P}$, then $\overline{\mathcal{S}}^{\mathcal{P}}$ denotes the longest temporal subsequence of \mathcal{S} such that $\overline{\mathcal{S}}^{\mathcal{P}} \preceq \mathcal{P}$. Finally, the coverage of \mathcal{S} by \mathcal{P} is defined by:

$$cov_{\mathcal{P}}(\mathcal{S}) = sim(\mathcal{P}, \overline{\mathcal{S}}^{\mathcal{P}}).$$

Example 3. If $\mathcal{P} = (A, [1, 3]), (B, [4, 5])$, and $\mathcal{S} = \{(A, [1.1, 3.4]), (D, [1.5, 5]), (B, [4.1, 5.3])\}$, then $\mathcal{S} \preceq \mathcal{P}$ and $\overline{\mathcal{S}}^{\mathcal{P}} = \{(A, [1.1, 3.4]), (B, [4.1, 5.3])\}$. The coverage is computed by:

$$\frac{(3 - 1.1) * (5 - 4.1)}{max(2 * 1, 2.3 * 1.2)} \approx 0.62$$

The temporal pattern coverage is null in two cases: (1) if \mathcal{P} does not cover \mathcal{S} , or (2) if at least one pair of corresponding temporal items do not overlap, i.e. $\exists i, [l_i^p, u_i^p] \cap [l_{f(i)}^s, u_{f(i)}^s] = \emptyset$.

Definition 9. (Temporal pattern strict-coverage) A temporal pattern \mathcal{P} strictly-covers a temporal sequence \mathcal{S} , denoted $\mathcal{S} \prec \mathcal{P}$, if $\mathcal{S} \preceq \mathcal{P}$ and $cov_{\mathcal{P}}(\mathcal{S}) \neq 0$.

3. Temporal pattern extraction

Temporal pattern extraction aims at finding the *interesting* temporal patterns from a set of temporal sequences (denoted \mathcal{D}). In our approach, interesting temporal patterns are the longest patterns such that sequences similar to this pattern can be frequently encountered in \mathcal{D} .

Definition 10. (Temporal pattern extraction) Let $\mathcal{D} = \{s_i\}_{i \in \mathbb{N}_m}$ be a set of m temporal sequences, and $f_{min} \in [0, 1]$ be the frequency threshold. Then, temporal pattern extraction consists in finding out the longest (i.e. with the highest dimension) temporal patterns $\{P_i\}_{i \in \mathbb{N}_p}$ such that for each P_i the examples strictly-covered by P_i are frequent:

$$\frac{|\{s \in \mathcal{D} \mid s \prec P_i\}|}{m} \geq f_{min}.$$

Example 4. (Temporal pattern extraction) Given the database \mathcal{D} that contains the following temporal sequences:

$$\begin{aligned} s_1 &= (A, [1, 3]), (B, [4, 5]) \\ s_2 &= (A, [1.2, 3.3]), (B, [3.8, 4.6]), (C, [5.8, 6.7]) \\ s_3 &= (A, [1.1, 3.4]), (D, [1.5, 5]), (B, [4.1, 5.3]) \\ s_4 &= (A, [0.9, 2.6]), (B, [4.1, 5.2]), (C, [6.2, 7.3]) \\ s_5 &= (A, [0.1, 0.8]), (B, [4.2, 5.2]) \end{aligned}$$

Given a frequency threshold $f = 0.5$, there is only one frequent symbolic signature of dimension 2: (A, B) . Then, temporal extents must be associated with each event, but several temporal patterns are possible:

- $((A, [0.1, 3.4]), (B, [3.8, 5.3]))$ strictly-covers all the examples, but the mean coverage is low (≈ 0.36).
- $((A, [1.1, 3.1]), (B, [4, 5]))$ has a better mean coverage (≈ 0.71), but does not strictly-cover s_5 .

Moreover, several temporal extents that strictly-cover the same set of temporal sequences can be computed for a temporal pattern. Our aim is to define a unique and representative temporal extent for each interval of a temporal pattern. Since, the distributions of temporal intervals can highlight such *representative* temporal extents, we propose a solution to construct unique and representative temporal extents which relies on the estimation of the interval distributions.

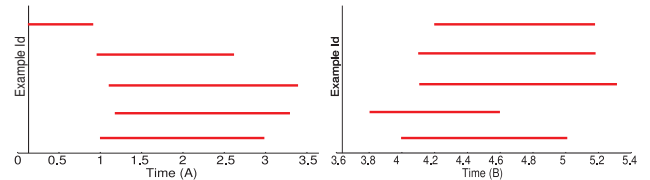


Figure 3. Interval distribution of A (left) and B (right).

For example, Figure 3 shows the distribution of A intervals and B intervals of the example 4 that can be used to extract the temporal extents. Clearly, temporal sequence s_5 rises a problem: all the sequences seems “similar” with respect to the B -intervals distribution, but s_5 does not seem “similar” to the others with respect to the A -interval distribution. Consequently, the distribution of A -intervals in

the pattern having (A, B) as symbolic signature can not be inferred from all A -intervals. More generally, this shows clearly that the interval distributions of a pattern can not be inferred from the interval distributions of its subpatterns. Therefore, interval distribution extraction is intertwined with symbolic signature construction. This is the temporal pattern extraction issue to tackle.

At this point, the hyper-cube representation is helpful. This representation allows to consider the distribution of the intervals for all temporal dimensions globally at the same time. In Figure 4, the frequency of temporal regions is outlined with darkening color: the darker a region is, the more temporal sequences overlap this region.

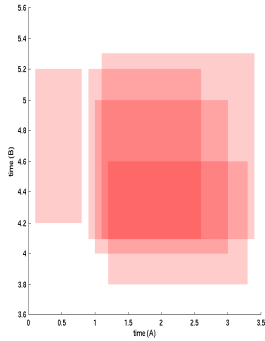


Figure 4. Distribution of temporal sequences (A, B) represented by hyper-cubes.

Our proposition is based on hyper-cube distributions which can be approximated using density-based multidimensional clustering (e.g. EM algorithm [7]). Then, extracted clusters are used to compute representative temporal bounds.

In our framework, we approximate distributions with Gaussian mixture models. Once computed, each Gaussian is associated with a possible temporal pattern. The temporal intervals to overlay over the symbolic signature are extracted from a Gaussian of the model by computing the bounds associated with an interval representing some ratio of the whole confidence. This is the *temporal extent threshold*. In our experiments, we have chosen 95% confidence intervals.

Figure 5 illustrates the Gaussian distribution estimation and the extracted temporal bounds. Two clusters represented by ellipses are computed, so two temporal patterns are extracted. Each ellipsis axis gives the temporal bounds for the item associated with the axis dimension.

Note that it is easy to understand 2-dimensional temporal sequence graphs and their distribution. For higher dimensions, graphical representations are almost impossible to understand but the same reasoning can be done.

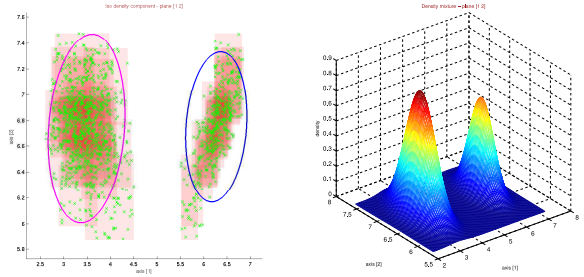


Figure 5. On the left, temporal sequences instances and Gaussians iso-density ellipsis. On the right, the density estimation.

4. Framework and algorithms

Our framework is based on the *APriori* algorithm [1] to construct the longest frequent symbolic signatures. The main idea of this algorithm is based on the following property: *every symbolic signature that covers a frequent symbolic signature is frequent*. This property is helpful to prune a lot of candidates of size n knowing the infrequent symbolic signatures of size strictly inferior to n because *if a symbolic signature of size n is covered by an infrequent symbolic signature then it is not frequent*.

The algorithm, whose pseudo-code is given in Algorithm 1, iterates two stages, namely (1) the selection of frequent temporal patterns of dimension n , and (2) the generation of symbolic signature candidates of dimension $n + 1$ from those of dimension n . These two stages are detailed in the following sections. The process terminates when there is no more potential longer candidate or when the candidates are no longer sufficiently frequent.

The algorithm has been adapted to process hyper-cube representations of temporal sequences. In particular, the selection of frequent temporal patterns uses the distribution estimation approach to extract the temporal intervals (see section 4.2), and the candidate generation uses *a priori* constraints enriched by temporal bounds (see section 4.3).

4.1. Main algorithm

The first step of the algorithm consists in constructing the frequent symbolic signatures of dimension 1. If an item (e.g. (A)) has less occurrences than $f_{min} * |\mathcal{D}|$, then it is not frequent and according to the *APriori* algorithm principle, it should not be taken into account. Then we enter in the main selection-generation loop. Assume that L_n , the set of symbolic signatures of dimension n , has been constructed. As will be seen later, these symbolic signatures are good candidates to be frequent. The selection phase

aims (1) at verifying the frequency of generated candidates and (2) at constructing their associated temporal intervals. Given $p \in L_n$, we first verify that the symbolic signature of p is frequent. Then, and only then, we can expect to find out temporal intervals using the density estimation of the interval distribution. The candidate selection function is detailed in the next section. This function returns a set of frequent temporal patterns that are added to M_n . Note that a symbolic signature can give rise to several temporal patterns (see Figure 5 for instance).

Once all symbolic signatures have been processed and if M_n is empty, then no temporal patterns of size n is frequent. Thus, the result given by the algorithm is the set of frequent temporal patterns of size $n - 1$.

Algorithm 1: Temporal pattern mining algorithm (inspired from the *APriori* algorithm).

Input: f_{min} : minimal support, \mathcal{D} : temporal sequences database, d : random points density

Output: M_n : the longest frequent temporal patterns (of dimension n)

// Construct frequent symbolic signatures of dimension 1 in L_1

$L_1 = \text{construct_L1}(\mathcal{D}, f_{min});$

$n = 1;$

while L_n is not empty **do**

$M_n = \emptyset;$

for $p \in L_n$ **do**

 // Selection phase

$f = \text{frequency}(p);$

if $f \geq f_{min}$ **then**

$M_n = M_n \cup \text{selection}(f_{min}, \mathcal{D}, p, f, d);$

end

end

if M_n is empty **then**

return $M_{n-1};$

end

 // Generation phase

$L_{n+1} = \text{generateCandidates}(M_n);$

$n = n + 1;$

end

return $M_n;$

4.2. Candidate selection

Algorithm 2 details the candidate selection function. This function aims at identifying the frequent temporal patterns in \mathcal{D} that can be associated with some symbolic signature p . There is two main tasks: (1) construct the representative temporal bounds for p , and (2) prune the infrequent temporal patterns.

The main part of the algorithm consists in estimating the density function for the distribution of the hyper-cube representation of p instances. We use a Monte-Carlo approach

to estimate the density function. For each sequence $s \in \mathcal{D}$ that is covered by p , random points (vectors in dimension n) are uniformly generated inside the hyper-cube defined by \bar{s}^p . The number of points is proportional to the volume of the hyper-cube. Moreover, to deal with the curse of dimensionality [2], this number is exponentially proportional to the dimension of the hyper-cube. The required number of points is given by the following formula:

$$nb = \text{vol}(\bar{s}^p) * d^{\text{dim}(p)} \quad (1)$$

where d is a parameter of our algorithm that can be easily adjusted depending on the time values expected in intervals.

Algorithm 2: selection($f_{min}, \mathcal{D}, p, f, d$)

Input: f_{min} : the minimal support, \mathcal{D} : temporal sequences database, p : symbolic signature, f : frequency of p , d : random points density

Output: M : set of frequent temporal patterns

// Generation of representative points for interval distribution

for $s \in \mathcal{D}$ **do**

if p covers s **then**

 // nb: number of points to generate

$nb = \text{vol}(\bar{s}^p) * d^{\text{dim}(p)};$

 add nb random points inside \bar{s}^p to hh ;

end

end

// Extraction of frequent temporal patterns from hh

$\text{GaussianMixtureModel} = \text{EM}(hh);$

for $gm \in \text{GaussianMixtureModel}$ **do**

if $f * gm.prop \geq f_{min}$ **then**

$p_{new} = \text{temporal_pattern}(p, gm);$

$M = M \cup \{p_{new}\};$

end

end

return $M;$

The distribution of generated points, denoted hh in Algorithm 2, represents the distribution of the intervals. Then, algorithm EM [7] is used to estimate the Gaussian mixture model from these points. In the most general case, the density function is:

$$f(\mathbf{x}) = \sum_{k=1}^N c_k (2\pi)^{-d/2} |\Sigma_k|^{-1/2} e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^t \Sigma_k^{-1} (\mathbf{x}-\mu_k)}$$

where N is the number of Gaussians, μ_k is the mean vector and Σ_k is the variance matrix of the k^{th} Gaussian, and c_k is the proportion of the Gaussian in the mixture.

In the EM algorithm the number of Gaussians N must be given, but the correct number of clusters is unknown.

Consequently, the algorithm is executed for several numbers of clusters and the best mixture is selected. In practice, the normalized entropy criterion (NEC) is used to select the number of clusters between 1 and $nb^{0.3}$ [4].

In addition, we assume that the intervals in a temporal pattern are independent. Consequently, the EM algorithm is constrained to extract Gaussians with diagonal variance matrices. This assumption reduces the computation cost.

Once, the Gaussian mixture model has been computed, the temporal patterns can be extracted. First of all, a Gaussian gm must have a sufficient proportion denoted c_k in the mixture formula and denoted $gm.prop$ in Algorithm 2. It means that if a symbolic signature has a frequency of f , then $f * gm.prop$ gives the number of temporal sequences strictly-covered by the temporal pattern associated with an extracted Gaussian gm . Such a temporal pattern is retained only if $f * gm.prop \geq f_{min}$.

Finally, the interval bounds are computed by:

$$\left[\mu_k - \frac{\alpha}{2}(2\pi)^{-d/2}|\Sigma_k|^{-1/2}, \mu_k + \frac{\alpha}{2}(2\pi)^{-d/2}|\Sigma_k|^{-1/2} \right]$$

where α is a parameter used to adjust the width of the interval (currently fixed to 0.8).

Algorithm 3: generateCandidates(M_n)

Input: M_n : set of frequent temporal patterns of dimension n

Output: L_{n+1} : set of candidate temporal patterns of dimension $n + 1$

// Generation of candidates

for $s \in M_n$ **do**

for $i \in L_1$ **do**

$P = \{s, (i, [-\infty, +\infty])\}$;

$L_{n+1} = L_{n+1} + P$;

end

end

// Suppression of a priori infrequent candidates

for $s \in L_{n+1}$ **do**

if s is a priori *infrequent* **then**

$L_{n+1} = L_{n+1} - s$;

end

end

return L_{n+1} ;

4.3. Candidate generation

Now, we present the candidate generation (see Algorithm 3). To generate only the potentially useful candidates, we extend the classical *APriori* constraint on symbolic signature to a constraint that takes into account the temporal intervals thanks to the hyper-cube representation. Let $P = \{(s_i, [l_i, u_i])\}_{\mathbb{N}_{n+1}}$ be a temporal pattern of dimension $n + 1$. P may be frequent if every orthogonal projection

of its hyper-cube on each subspace of dimension n overlaps with at least one hyper-cube of M_n , the set of frequent temporal patterns of dimension n , *i.e.*:

$$\forall l \in [1, n + 1], \exists p \in M_n, \text{sim} \left((s_i, [l_i, u_i])_{\mathbb{N}_{n+1} - \{l\}}, p \right) \neq 0$$

In the first phase of the generation, temporal patterns of dimension $n + 1$ are generated by concatenating an element of M_n and an element of L_1 (the set of frequent items). The element of L_1 is associated with infinite temporal bounds. In a second phase, the *a priori* infrequent elements are pruned.

5. Evaluation

The algorithms have been implemented in MatLab v7.6 (R2008a)¹. We have used the MixMod library [3] to compute Gaussian mixture models. Our experiments were performed on a bi-processor Pentium IV 3.6GHz, 2Go of RAM. To evaluate the properties of our algorithm, we have used synthetic data sets. The simulation process is flexible enough to quantitatively evaluate several properties such as accuracy, computation time and robustness against the noise.

To our knowledge, there is no other algorithms for extracting temporal sequences with quantitative temporal intervals that we could quantitatively compare with. In section 6, we present a qualitative comparison of our temporal pattern extraction method to other temporal sequence mining methods.

5.1. Data set simulation

The data simulation process generates random temporal sequences based on a *temporal pattern prototype* to which is added temporal noise and structural noise.

A *temporal pattern prototype* is a set of n quintuplets $(\{(E_i, \mu_{b_i}, \sigma_{b_i}, \mu_{d_i}, \sigma_{d_i})\}_{i \in \mathbb{N}_n})$ where E_n is an event name, μ_{b_n}, μ_{d_n} (resp. $\sigma_{b_n}, \sigma_{d_n}$) are the mean (resp. the standard deviation) of the generated begin instants and **durations** of E_n . This prototype specifies the temporal pattern to discover $(\{(E_i, [\mu_{b_i}, \mu_{b_i} + \mu_{d_i}])\}_{i \in \mathbb{N}_n})$, and the standard deviation parameters quantifies the Gaussian distribution of the interval bounds. The larger the standard deviation is, the harder the temporal pattern extraction is. For the sake of simplicity, all standard deviations are equal to $tN \in [0, 1]$. This parameter quantifies the *temporal noise* in the data set.

In addition, some *structural noise* can be added. A structural modification consists in randomly deleting an event or modifying an event name. $sN \in [0, 1]$ is the ratio of structurally modified temporal sequences in the data set. Finally,

¹Source code for the algorithm and the data generator available on-line: <http://www.irisa.fr/dream/QTempIntMiner/>.

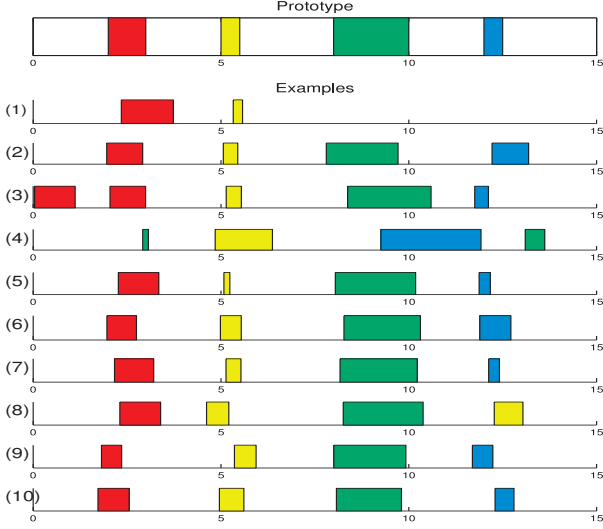


Figure 6. Prototype (upper graph) and 10 generated examples.

totally randomly generated temporal sequences are added to the data set with ratio $rP \in [0, 1]$.

Figure 6 illustrates the temporal sequence generation process ($nb = 10$, $sN = 0.1$). The fourth sequence has been randomly generated ($rP = 0.1$), several sequences such as sequence number 8, have been structurally corrupted ($sN = 0.1$) and time bounds fluctuates due to the temporal noise ($tN = 0.2$).

In the following, the prototype used to generate the data sets ($nb = 20$) was $\{(E_k, [3k, 3k + 1])\}_{k \in \mathbb{N}_n}$. When not explicitly given, the parameters take their default values $n = 4$, $nb = 20$, $sN = 0$ and $tN = 0.2$. For each set of parameters, 10 different data sets were generated to yield statistically significant results.

The accuracy (in $[0, 1]$) of the temporal pattern extraction process is evaluated by the similarity value between the prototype and the best extracted temporal pattern (see Definition 6). This means that if the temporal pattern and the prototype share the same symbolic signature, then the accuracy is given by the relative common volume. Otherwise the accuracy will be set to 0. A non-zero accuracy points to a correct symbolic signature extraction (this is always the case with the basic parameters), thus the higher the accuracy, the better the extracted temporal bounds.

5.2. Experiments and results

The results are organized as follow: first, we present evaluations of the computation time and the accuracy of the algorithm according to the parameters nb , n , tN and sN (linked to the temporal sequence representation). Then we

demonstrate the robustness of the algorithm in finding out the interesting temporal patterns in a large amount of temporal sequences. Finally, we highlight some properties on examples.

Computation time and accuracy Short computation time is a strong constraint on sequential pattern mining algorithms. The computation time of *APriori* is known to increase exponentially with the pattern size (*i.e.* the prototype dimension n). Moreover temporal bounds extraction in the selection phase is also time consuming. The number of examples for the EM algorithm is proportional to d^n , where d is a parameter (see *eq.* 1). Practically, the maximum dimension of the extracted temporal pattern may be limited to have reasonable computation times.

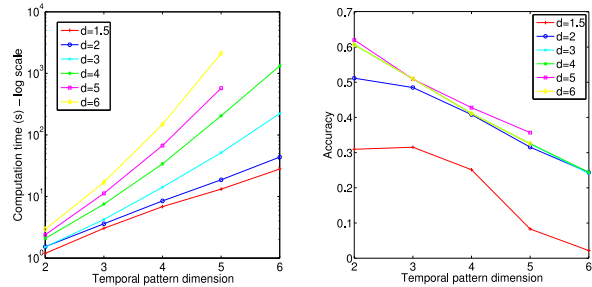


Figure 7. Computation time (on the left) and accuracy (on the right) w.r.t. the prototype size.

The results (Figure 7, left) show that the computation time increases exponentially with the prototype dimension (n ranges from 2 to 5). For $d = 4$, the computation time ranges from 2'' to 3'41''. Figure 7 right gives the accuracy. The accuracy decreases notably when the dimension increases. This comes from the fact that the accuracy is evaluate from the similarity: by definition the similarity computes the product of the relative sizes (lying in $[0, 1]$) of the intersection along each dimension of two hyper-cubes. Experiments have been performed with six values of d from 1.5 to 6. We expected that the computation time should increase with d , while the accuracy should become better. Practically, this is the case for $d \leq 3$. However, we did not notice any statistically significant improvement of the accuracy for $d \geq 3$. Nonetheless, the computation time is significantly higher, so we set $d = 3$ for the other experiments.

Figure 8 illustrates the computation time and the accuracy with respect to the number of examples (nb ranges from 100 to 1000). On the one hand, the computation time increases linearly, and on the other hand, the accuracy remains quite the same. Thus, this experiment shows that our

algorithm is robust with respect to the database size.

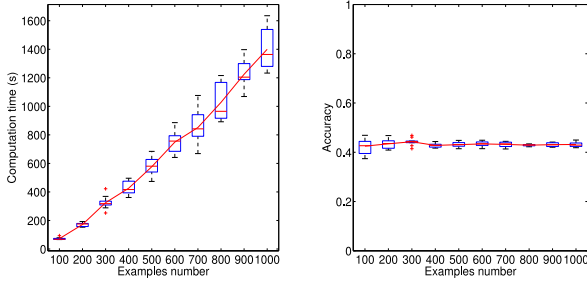


Figure 8. Computation time (on the left) and accuracy (on the right) w.r.t. the examples number. Boxplots show the variances.

Finally, Figure 9 illustrates the accuracy with respect to the temporal noise. There is two parts in this graph: at the beginning, the accuracy increases, then it decreases when more temporal noise is added. The second part shows a normal behavior: the higher the temporal noise, the less accurate the temporal intervals. The dashed line highlights failures in pattern extraction (the symbolic signature could not be successfully extracted). This means that when the temporal variances become too high, the algorithm is no more able to reconstruct the symbolic signatures due to the difficulty to identify the correct Gaussian mixture models. For $tN = 0.6$, it occurs only in 10% of the experiments, but for $tN = 1$ it occurs in 50% of the experiments.

We expected to obtain a very good accuracy and low variances when the temporal noise is low. But the first part of the Figure 9 shows the opposite. It can be explained by observing that many extracted intervals are shifted compared to the prototype intervals. This behavior is explained below.

We conclude from these experiments that (1) the algorithm extracts temporal patterns with accurate temporal intervals, and (2) in spite of the theoretical drawbacks (confirmed by the experimental results), the computation time is reasonable.

Temporal interval enlargements and shifts For temporal interval extraction, we hypothesize that the interval distribution fits a Gaussian distribution. We identified two cases where this hypothesis is false. The first one occurs when the variance of the lower (or upper) bound of a prototype interval is high compared to the duration of this interval (see Figure 10, left). In this case, the extracted temporal interval is larger than expected.

The second case occurs when the variance of the lower bound of a prototype interval is low compared to the variance of its duration (see Figure 10, right). In this case, the

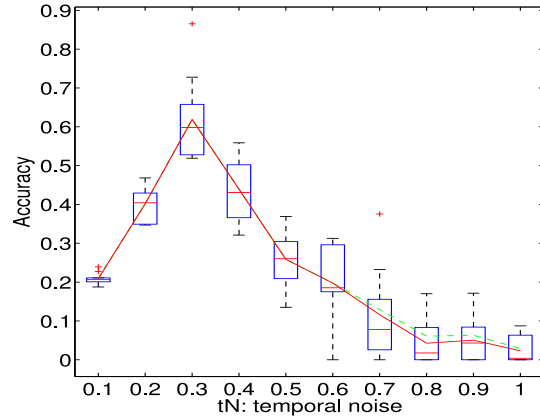


Figure 9. Accuracy w.r.t. temporal noise. The dashed line gives the accuracy mean computed with non-zero accuracies.

extracted temporal interval is shifted. In fact, the distribution is not Gaussian anymore: it is rather exponential.

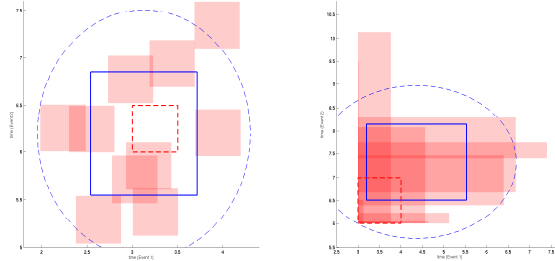


Figure 10. Two cases of inadequate interval extraction. Plain line rectangles show the temporal bounds extracted from the computed Gaussians (ellipses). Dashed line rectangles show the expected temporal bounds.

Knowing these possible behaviors is helpful to determine the assumption under which the extraction algorithm will be efficient. As a consequence, the data set must be such that the variance of interval lower bounds must be *low* but *sufficiently high* compared to their respective duration variance.

Robustness to structural and database noises In the previous experiments, database temporal sequences were corrupted with temporal noise only. In the present section, we keep some temporal noise ($tN = 0.2$) but we introduce structural noise in the temporal sequence generation. The robustness to structural corruption can be assessed by varying the f_{min} parameter of the *APriori* framework. As-

suming that $rP = 0$, *i.e.* that there are only positive examples (possibly corrupted) of the temporal pattern and, given f_{min} , we expect that the algorithm will be robust with respect to a structural noise equal to f_{min} .

Figure 11 illustrates the accuracy with respect to the structural noise level and $f_{min} = 0.5$. Temporal pattern extraction failures occur when $sN \geq 0.35$ (only 1 failure out of the 10 trials for $sN = 0.35$). When $sN = 0.5$ temporal pattern extraction fails in more than half of the cases (6 failures out of 10 trials). This experiment confirms that when f_{min} is low the robustness of our algorithm against the structural noise is high.

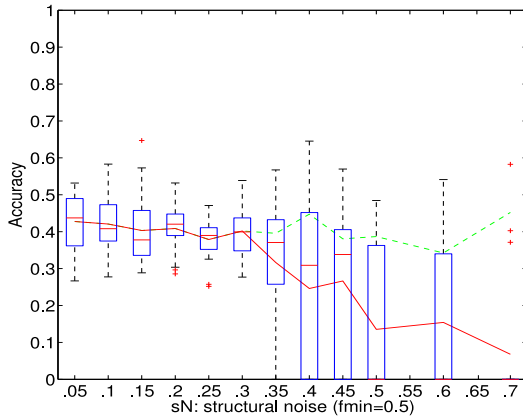


Figure 11. Accuracy with boxplots w.r.t. structural noise. The dashed line gives the mean of non-zero accuracies.

However, in real databases, positive examples of a temporal pattern are mixed (1) with positive examples of other temporal patterns and (2) with random irrelevant temporal sequences. Then, f_{min} must be sufficiently low both to find out the few positive examples of a temporal pattern, and to be robust against the structural noise. But, if f_{min} is too low, the computation time may hugely increase and irrelevant patterns could be extracted.

To confirm this hypothesis, we have generated a more realistic data base containing 1000 temporal sequences with 80% of randomly generated sequences ($rP = 0.8$). Two different prototypes were used to generate relevant temporal sequences (10% each). Consequently, f_{min} must be lower than 0.1 (ratio of positive examples in the database per temporal pattern) to consider relevant temporal sequences as frequent, but due to additional structural noise ($sN = 0.2$), it must be lower than $0.1 * (1 - sN)$. Thus we choose $f_{min} = 0.05$. The experiment showed that our algorithm is able to find out at least one temporal pattern and the two temporal patterns in 65% of the trials with a mean accuracy of 0.33 and a mean computation time of 9'07''.

Additional properties Other experiments have been performed to illustrate interesting additional properties of our algorithm. For example, our algorithm can extract (1) temporal patterns with temporal items that overlap or (2) temporal patterns with twins, *i.e.* several identical events (associated with different intervals). Contrary to existing approaches, *e.g.* [8], these temporal patterns do not introduce difficulties to our algorithm thanks to the hyper-cube representation that separates the temporal information for each item. In the case of temporal patterns with twins, two dimensions can be separated if two representative temporal extents are extracted. This can be computed from Gaussian mixture models. For example, it can distinguish the two representative Gaussians of Figure 5.

6. Related work

Temporal sequence mining involves two different kinds of issues: FSPM algorithm with time constraints [14, 15], and temporal relation discovery [9, 11, 12, 17]. Hirate *et al.* [10] propose a generalized FSPM with the capability to handle two kinds of item interval measurements, item gap and time interval.

Time constraints approaches, make use of minimum and maximum gap values that must be defined as the gap constraint by the user. Then the algorithms, such as SeqLog [14] or GTC (Graph for Time Constraints) [15], extract ordered sequences of items that respects these gaps. However, the algorithm do not discover the time constraints. The gaps are used to get rid of insignificant patterns and rules so as to reduce the number of generated patterns and rules.

Temporal relation discovery approaches use extended definitions of temporal sequences. In [11, 13], qualitative temporal relations on intervals, *i.e.* Allen temporal relations, are associated with sequence items. Höppner [11] propose a level-wise algorithm inspired from the *APriori* algorithm to extract efficiently both event sequences and temporal relations.

Chen *et al.* [5] or Quiniou *et al.* [16] propose an intermediate representation of temporal sequences between qualitative temporal relations and quantitative temporal constraints. Chen *et al.* [5] uses the concept of fuzzy sets to extend the original *APriori* algorithm and PrefixSpan algorithm so that fuzzy time-interval sequential patterns are discovered from databases. Quiniou *et al.* [16] uses a first-order relational learning method to construct a graph of temporal constraints called *chronicles*.

The closest approaches to ours are [8, 9, 12, 17]. The temporal information associated with temporal sequences is represented by quantitative interval extents. In [9], Giannotti *et al.* propose an extension of the sequence mining paradigm to linear *temporally-annotated* sequences, where each transition $A \rightarrow B$ in a sequence is annotated with

a typical transition time t , denoted by $A \xrightarrow{t} B$. The notion of *Delta-Pattern*, proposed by Yoshida et al. [17] is quite similar to temporally annotated sequential patterns. *Delta-Patterns* have the form $A \xrightarrow{[0,3]} B$, denoting a sequential pattern $A \rightarrow B$ that frequently occurs in the data set with transition times from A to B that are contained in $[0, 3]$. The FACE system [8] adapted the *APriori* framework [1] to extract the temporal extents of the interval between two events as the enveloping interval of the temporal sequence instances intervals. As us, Giannotti et al. and Yoshida et al. [17] use a clustering method to extract the representative values of the transitions. Yoshida et al. [17] provide an heuristic for finding some frequent *Delta-Patterns* but their method is not exhaustive. Giannotti et al. [9] propose to combine a candidate sequence generation step with a clustering step, but do not provide details nor evaluations of their method. In both methods, the main issue of intertwining a pattern growing method with temporal interval generation based on the analysis of interval distributions is not explicitly addressed.

7. Conclusions

In this paper we introduce a temporal extension of the classical sequential pattern mining which adds quantitative temporal information to patterns. Using a hyper-cube representation of temporal sequences, the temporal extents of pattern events can be globally extracted. We propose an algorithm that solves the issue of intertwining the interval distribution extraction and symbolic signature construction.

Our algorithm is based on the *APriori* algorithm and an approximation of the interval distribution. On the one hand, a variant of the *APriori* algorithm enables the extraction of the longest patterns in large temporal sequence database. On the other hand, the use of Gaussian mixture models for approximating the interval distributions enables the extraction of the representative quantitative temporal extents of a pattern.

As *APriori* our method is exponential with the size of the patterns but it remains linear with the number of sequences in the database. The practical efficiency of the method has been evaluated on synthetic data sets. The results show that while the pattern growth technique makes our algorithm robust against the structural noise, the use of Gaussian mixture models makes it robust against temporal noise.

References

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering*, 1995.
 [2] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[3] C. Biernacki, G. Celeux, G. Govaert, and F. Langrognnet. Model-based cluster and discriminant analysis with the MIXMOD software. *Computational Statistics and Data Analysis*, 51(2):587–600, 2006.
 [4] H. Bozdogan. Choosing the number of component clusters in the mixture-model using a new informational complexity criterion of the inverse-Fisher information matrix. In *Proceedings 16th Annual Conference of the GfKI*, pages 40–54, 1993.
 [5] Y.-L. Chen and T. C.-K. Huang. Discovering fuzzy time-interval sequential patterns in sequence databases. *IEEE Transaction on Systems, Man and Cybernetics*, 35(5):959–971, 2005.
 [6] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining*, pages 493–498, 2003.
 [7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.
 [8] C. Dousson and T. Duong. Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 620–626, 1999.
 [9] F. Giannotti, M. Nanni, D. Pedreschi, and F. Pinelli. Mining sequences with temporal annotations. In *Proceedings of the Symposium on Applied Computing*, pages 593–597, 2006.
 [10] Y. Hirate and H. Yamana. Generalized sequential pattern mining with item intervals. *Journal of computers*, 1(3):51–60, 2006.
 [11] F. Höppner. Learning dependencies in multivariate time series. In *Proceedings of the Workshop on Knowledge Discovery in (Spatio-)Temporal Data*, pages 25–31, 2002.
 [12] S.-Y. Hwang, C.-P. Wei, and W.-S. Yang. Discovery of temporal patterns from process instances. *Computers in Industry*, 53(3):345–364, 2004.
 [13] P.-S. Kam and A. W.-C. Fu. Discovering temporal patterns for interval-based events. In *Data Warehousing and Knowledge Discovery (DaWaK)*, pages 317–326, 2000.
 [14] S. D. Lee and L. De Raedt. *Database support for data mining applications : discovering knowledge with inductive queries*, chapter Constraint based mining of first order sequences in SeqLog, pages 154–173. 2004.
 [15] F. Massegli, P. Poncelet, and M. Teisseire. Efficient mining of sequential patterns with time constraints: Reducing the combinations. *Expert Systems With Applications*, 40(3):to appear, 2008.
 [16] R. Quiniou, G. Carrault, M.-O. Cordier, and F. Wang. Temporal abstraction and inductive logic programming for arrhythmia recognition from electrocardiograms. *Artificial Intelligence in Medicine*, 28:231–263, 2003.
 [17] M. Yoshida, T. Iizuka, H. Shiohara, and M. Ishiguro. Mining sequential patterns including time intervals. In *Proceedings of the conference on Data Mining and Knowledge Discovery: Theory, Tools and Technology II*, pages 213–220, 2000.