Enriching Multivariate Temporal Patterns with Context Information to Support Classification

Frank Höppner, Sebastian Peter and Michael R. Berthold

Abstract In this paper we consider classification tasks where the class depends on the co-evolution of multiple variables over time, for instance, "if A happens before B and in the meantime we do not observe C, then we have a failure of class X". We present a two-phased approach to derive such patterns from data. In the first step, we seek the most specific pattern that still matches all data from one class and in the second step we constrain the pattern further, such that it discriminates with respect to other classes. While the second step is directly motivated by the classification task, the first step enables the user to better match his or her mental model of the temporal process to the patterns derived by the classifier. The experimental evaluation on the libras dataset has shown that the additional first step not only improves the interpretability, but also the classification results.

1 Introduction

Measuring and recording data is easy and cheap nowadays, but in some applications substantial conclusions can only be drawn if we extend our observations to a certain period of time. An operator who is controlling a chemical production process, a user interacting with a technical device, a medic administering a drug to a patient – in all these cases instantaneous information does not help to differentiate between

Frank Höppner
Ostfalia University of Applied Sciences, Department of Computer Science, D-38302 Wolfenbüttel, Germany e-mail: f.hoeppner@ostfalia.de

Sebastian Peter
University of Konstanz, Nycomed-Chair for Bioinformatics and Information Mining, Box 712, D-78457 Konstanz, Germany e-mail: sebastian.peter@uni-konstanz.de

Michael R. Berthold
University of Konstanz, Nycomed-Chair for Bioinformatics and Information Mining, Box 712, D-78457 Konstanz, Germany e-mail: michael.berthold@uni-konstanz.de

a successful and a failed process, decide about the ergonomics of a man-machine interface or judge about the chances of patient recovery. It is necessary to observe multiple attributes over a period of time to derive rules specifying how a class label may depend on the *history* of observations.

Measuring a couple of variables over a period of time turns the classification task into a high-dimensional problem. As classifiers seek for the *best attribute* to discriminate between the classes, they may eventually come up with classification rules that only depend on a few of these attributes. However, users wants to align the findings with their mental model, which is difficult if most of the temporal context is ignored or lost by the classifier. In this paper, we propose to include more background information by means of a temporal outline or sketch, which is then refined by the classifier. This approach tackles two problems: It reduces the danger of overfitting, because it reduces the possibilities of combining arbitrary features that may occur otherwise at any time in the recorded history and, secondly, it provides the necessary background information for the user when inspecting the result.

The remainder of the paper is organized as follows: In Sect. 2 we briefly discuss the representation of temporal data and review related work. The classifier we are going to use in this paper is reviewed in Sect. 3, while an approach to provide the aforementioned background information is discussed in Sect. 4. Results on the libras data set are discussed in Sect. 5. Section 6 finally concludes the paper.

2 Representation and Related Work

Rather than considering values individually, we employ temporal abstractions such as 'rising temperature', 'connection established', 'low user activity', or 'increased variance'. We thus describe the evolution by means of temporal predicates: denoting the temporal dimension by $\mathbb{T}$, a temporal predicate P_l is a function $P_l : \mathbb{T} \to \mathbb{B}$, where l is called the label of the predicate P. Examples for predicates (and especially their label) were given above. The choice of predicates is domain dependent and part of the feature selection step in data mining. A set of predicates (which we will call history H) may be depicted by plotting them against the temporal dimension (cf. Fig. 1). The use and visualization of *temporal abstractions* has a long tradition in the medical domain [9]. Note that in contrast to stream mining approaches, where a single but potentially infinite stream of data is considered, we assume that multiple (finite) labelled histories are available.

Various ways of defining patterns in a stream of labeled intervals have been proposed in the literature, many of them relying on Allen's interval relationships [1] (cf. Fig. 2) or variants thereof. Some approaches define a history by specifying the exact relationship for every pair of intervals [3], others allow for a set of possible relationships [4]. The representation by sequences of chords [6] uses a partially ordered sequence of simultaneous (sub)intervals to define a pattern. Other proposals consider a different set of interval relationships or specify the relationship between temporal intervals only partially [5].

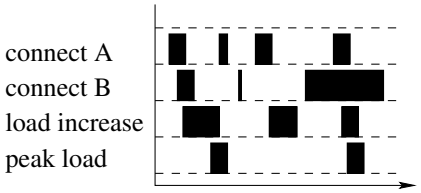

Fig. 1 Representation of Evolving Data: the black rectangles denote the intervals when the predicate holds (labels on the left).

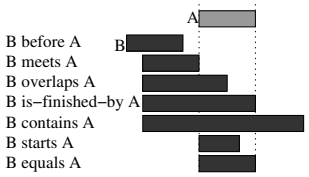

Fig. 2 Thirteen possible relationships between two intervals. The inverse relationships (before $\leftrightarrow$ after) have been omitted.

While these approaches have their individual strengths, they also have their weaknesses even when it comes to represent simple situations. Thinking of predicting a certain state of some network server (breakdown, overload, malfunction, etc.) on the history of, say, the last 24 hours, a situation as simple as "there was only one connection to server A" (during the last n hours) is usually prohibitive to discover using approaches based on association rules [3, 6], as they count occurrences of events and often rely on a quickly decreasing count of co-occurrences, which forbids an inclusion of *absent features* during counting. A situation like "at some point in time, both A and B hold" is a challenge to approaches such as [3], because they rely on explicitly specified interval relationships (which are ambiguous in this case). Temporal constraints "the connection to A was lost for at least 4 hours" or "... at most 4 hours" are usually ignored or introduced in a postprocessing step.

3 Representing and Classifying Temporal Data

To support an intuitive understanding we choose a rule-based approach where the conclusion part predicts the class and the premise of the rule contains a pattern that has to be matched to a given history. In [7] a notion of a pattern, called *template history*, has been introduced. A template history may be visualized as in Fig. 1, but this time a black box is understood as a constraint that has to be fulfilled by a matching history. The constraints on the presence of temporal abstractions are not fixed in time to compensate dilational and translational effects, only the order in which the constraints have to be fulfilled must be preserved. A template is thus decomposed into a number of n successive blocks whose absolute duration may vary from case to case. Together with a selection of m temporal predicates, we obtain an $m \times n$ matrix C where each cell $C_{i,j}$ represents a constraint on the i^{th} predicate in the j^{th} block (cf. Fig. 3). We distinguish between four different constraints:

Definition 1 (predicate constraint). Given a temporal interval $T \subseteq \mathbb{T}$ and a predicate P, we say (a) *P is present during T* if $\forall t \in T : P(t)$, (b) *P is absent during T* if $\forall t \in T : \neg P(t)$, (c) *$P$ exists during T* if $\exists t \in T : P(t)$ and (d) *P disappears during T* if $\exists t \in T : \neg P(t)$. If no condition is posed, we say *P is unconstrained during T*. By $\mathbb{C}$ we denote the set of constraints $\{$present, absent, exists, disappears, unconstrained$\}$.

Besides the constraints in the cells of the matrix, we may additionally constrain the duration of each block:

Definition 2 (template). A tuple $T = (L, n, C, D)$ is called a template if L is a set of labels, $n \in \mathbb{N}$, $C : L \times \{1,..,n\} \to \mathbb{C}$ and $D : \{1,..,n\} \to (\mathbb{T} \cup \{\infty\})^2$ satisfying $1 \leq d_{\min} \leq d_{\max}$ for any $D(i) = (d_{\min}, d_{\max})$, $1 \leq i \leq n$. The map C constrains the predicate in each block, the map D contrains the block duration.

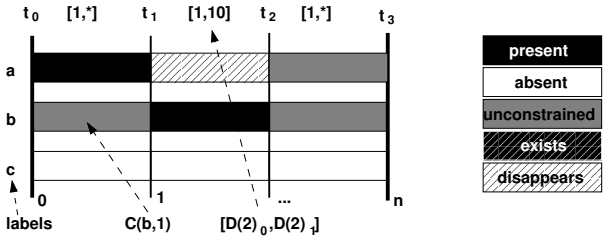

Fig. 3 Illustration of the template definition. The predicate constraints are color-coded.

Figure 3 shows an example template with $n = 3$ blocks, defined by four time points (vertical lines), where the leftmost and rightmost time point shall always represent the start and end of the history. The bottom row declares that a predicate P_c is absent in the whole history. Somewhere in the history (2nd block), P_b is present (P_b may be present or not elsewhere (=unconstrained)). P_a is present from the very beginning, but disappears while P_c is present in the 2nd block. The duration of the first block is arbitrary, the second block takes 1 to 10 time units ($D(2) = (1, 10)$), the last block may again have any (positive) duration ('*' represents '∞').

Matching a template to a history involves the determination of points t_i in time (temporal alignment) such that all constraints hold.

Definition 3 (match). Let $T = (L, n, C, D)$ be a template and H be a history. Let $[t_{\min}, t_{\max}]$ be the smallest interval subsuming $\cup_{P \in H} dom(P)$. T matches a history H if and only if (a) there is a predicate $P_l \in H$ for every $l \in L$, (b) there are $t_i \in \mathbb{T}$, $0 \leq i \leq n$, with $t_0 = t_{\min}$, $t_i \leq t_{i+1}$, $t_n = t_{\max}$, (c) for every $l \in L$ and $i \in \{1,..,n\}$ the constraint $C(l, i)$ holds for P_l within $[t_i, t_{i+1})$ and finally (d) for all $1 \leq i \leq n$: $t_i - t_{i-1} \in [d_{\min}, d_{\max}]$ with $(d_{\min}, d_{\max}) = D(i)$.

In [7] we proposed a method to explore the space of templates to find good discriminators between differently labeled histories. The search algorithm implements a general-to-specific search: It begins with an empty pattern and specializes it further to improve some measure of interestingness (we used the J-measure [10] as it balances the generality (applicability of the rule) and the interestingness (deviation from a priori knowledge)). The initial template that matches all histories consists of one block, all predicate constraints are *unconstrained* and so are the temporal constraints $(1, \infty)$. While a propositional rule can only be specialized by an additional condition (like *outlook*=*sunny*), there are at least three ways to specialize a

template: we may look at it in a finer resolution (by subdividing the temporal axis further), we may change or add a predicate constraint (for some label and block), or may introduce or change an existing temporal constraint. We thus have chosen three different specialization operators to address each of these aspects. The general idea for all refinement operators is to search for specializations that improve the measure of interestingness. A more detailed description of the operators and the quality of the learned patterns can be found in [7].

4 Providing Background Information

The idea of providing 'temporal context' in a template is to find some (most specific) pattern that matches all instances of a given class. Such a pattern may be used as a starting point for the beam search mentioned in Sect. 3. The problem of finding such a pattern is closely related to the alignment of multiple sequences, which is known to be NP-complete [11]. As all instances of the same class may in principle share a considerable number of blocks, the use of pattern mining algorithms that enumerate subpatterns is prohibitive because the number of subpatterns grows exponentially with the length of the sequence. However, we do not rely on the *optimal* or even the *most specific* pattern, but assume that any pattern that is shared by all instances of the same class will help to provide contextual background. Therefore we are duly satisfied with an approximate or heuristic solution to this problem. One possible approach will be discussed in the remainder of this section, but we do not claim any specific properties or advantages of this solutions: but our intention is to demonstrate that (any) common subpattern is potentially useful.

The idea behind our simple heuristic method is to exploit the fact that each instance itself should match the sought common subpattern – and that we thus may identify it by simplifying the instance subsequently. At first, an arbitrary selected history is transformed into a template history: Whenever a predicate changes its value, we introduce a new block. If the predicate holds during the block, we place a *present* constraint in the respective block, otherwise we leave it *unconstrained*. A copy of this template history is created where all predicate constraints are set to *unconstrained*, which is trivially matched by all instances. Next, we transfer the

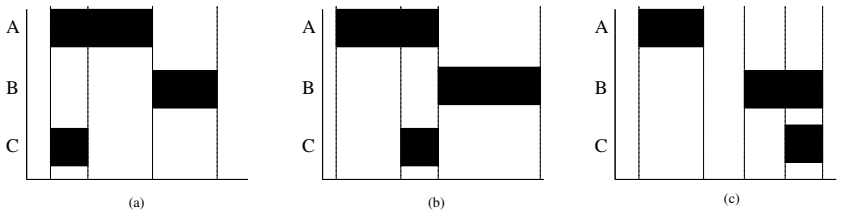

Fig. 4 Three sample histories for the starting pattern algorithm

present constraints (one by one) from the instance pattern to the (initially blank) copy and only keep it if it still matches all instances of its class.

For example: Given the three different histories shown in Fig. 4, we want to find a common subpattern, shared by all three histories. We start by using (a) as the start instance (please note that every instance could be picked). In the first step we create the template history by counting the blocks (segments between the dashed lines because there is at least one predicate change) and convert it into the history shown in Fig. 5. Finally we change all predicate constraints to *unconstrained* as shown in Fig. 6.

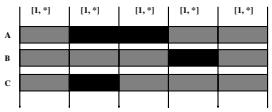

Fig. 5 Sequence (Fig. 4(a)) transformed into a template history.

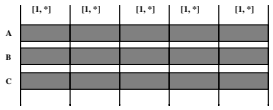

Fig. 6 Template history (Fig. 5) with all predicate constraints changed to *unconstrained*.

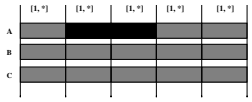

Fig. 7 Starting pattern after adding the A-Label intervals to the starting pattern in Fig. 6.

In the second step we add constraints to the pattern and check if the resulting pattern still matches all instances. Therefore we go through the original template history row by row and transfer the *present*-constraints to the pattern. Furthermore we add a new *unconstrained* block before and/or after the modified block to relax the required predicate relationships. We obtain four possible patterns in total – if more than one turns out to match all histories we choose the most specific one. For label A the algorithm adds the *present* constraints as shown in Fig. 7 directly to the pattern, so we inspect the refinements for label B in more detail. The four possible patterns shown in Fig. 8 are created as the possible refinements. Evaluating the first pattern shows that the instance in Fig. 4(c) is not matched anymore because the relation 'A *meets* C' is not present (but 'A *before* B'). The second pattern matches all sequences because the meet-relationship has been relaxed by the intermediate *unconstrained*-block. The remaining two patterns duplicate the final *unconstrained*-block but do not add any substantial differences. A further refinement is not possible, as there is no position for B that matches all three histories in Fig. 4.

Drawbacks of heuristic approach. From the example above we also recognize the drawback of this approach. The resulting patterns depend on the order in which the labels are added to the pattern. If we had started with adding a *present* constraint for predicate C we would not be able to add any more constraints, because all other intervals occur in different relationships to C.

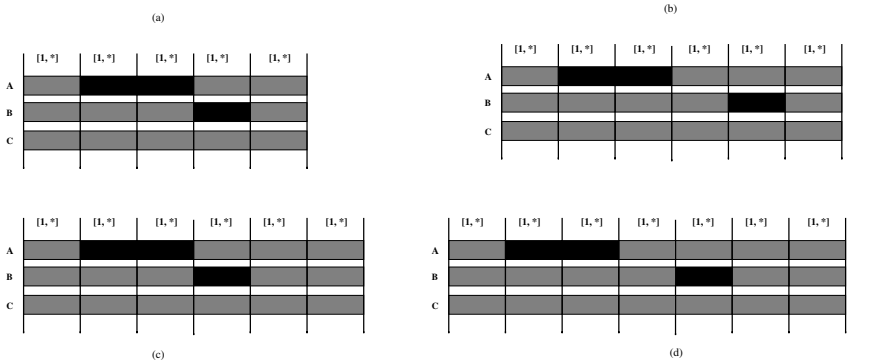

Fig. 8 Four template histories tested by the algorithm by adding B to the pattern shown in Fig. 7.

5 Experimental Evaluation

We applied our algorithm to the libras movement data set from the UCI repository [2]. It contains 15 different signs described by their characteristic hand movements over 45 time frames, where the current x- and y-positions of the hand were recorded. There are 24 instances per sign, 360 in total.

Data preperation & evaluation settings. In a first step we have manually inspected all hand movements and removed clear outliers and incomplete movements, such that not only parts of the hand movement appear in each class but the complete sign is visible. We have subsequently extracted predicates that represent the hand movement, e.g., the speed of the movement (overall speed and separate movement in x- and y-direction). We used a priori defined thresholds and the following labels only:

- x-movement: fast left ($--$), left ($-$), constant (o), right (+), fast right (++).
- y-movement: fast down ($--$), down ($-$), constant (o), up (+), fast up (++)
- x/y-moveall: fast ($++$), normal ($-$) (this label is the same as x/y-movement without distinguishing between left/right resp. down/up).
- Curve: nearly same direction (o), middle change of direction ($+$), abrupt change of direction($++$)

For example, a fast hand movement to the upper left may be recognized by observing predicates x-movement $--$ and y-movement $++$ at the same time.

We divided the preprocessed data into training (66%) and test (33%) data. For each sign we constructed a shared pattern and refined it using the classifier in [7] on the training set. The signs #10 and #12 were merged to just one class #10, because the set of features we had chosen was not suited to distinguish between these two hand movements. For the evaluation against the test set, we matched an instance against all obtained patterns – if an instance matches only a single rule pattern, the classification rule predicts the class; if no unique pattern matches, we classify it as "cannot predict".

5.1 Effect on Interpretability

Before discussing the classification performance we start by comparing the learned template histories for three different hand movements shown in Fig. 9, 10 and 11. We want to demonstrate the usefulness of the identified 'common pattern' for aligning it with the user's mental model of the considered process.

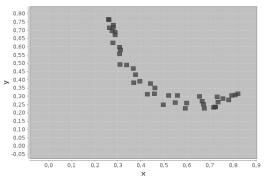

Fig. 9 Example hand movement for sign #1.

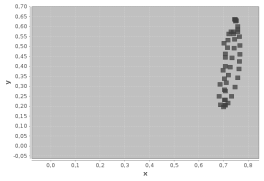

Fig. 10 Example hand movement for sign #3

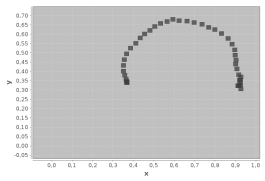

Fig. 11 Example hand movement for sign #14.

Sign #1. The pattern for sign #1 found without a start pattern is shown in Fig. 12. By inspection of this pattern, it is quite difficult to recover the actual hand movement, because it consists of many *absent*-constraints, which are difficult to align with a mental model of the hand movement. Furthermore, the first and last blocks consist of *unconstrained*-conditions only. Thus we do not get any information about what may happen *before* or *after* the pattern or at what time it might occur.

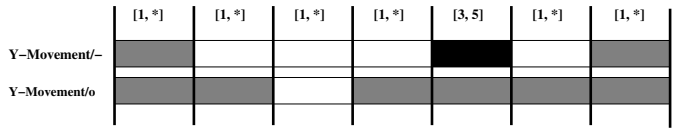

Fig. 12 Pattern found by the beam search without a starting pattern for sign #1.

Fig. 13 shows the pattern found with the help of a starting pattern: the many *present*-constraints support the user in understanding the actual hand movement. The pattern describes the hand movement almost completely: a fast left-right-left movement (*present* x-movement/–, x-movement/++ and x-movement/–) combined with an up-down-up movement (*present* y-movement/+, y-movement/- and x-movement/+). We also observe periods of 'high speed movements' and low speed when changing directions. In total the key features of the sign #1 as shown in Fig. 9 are well reflected.

Sign #3. Fig. 14 shows the pattern learned for the sign #3 without a starting pattern. It is again a simple pattern which forbids the occurrence of a fast left or right movement (*absent x-moveall/++*) at any time and requires a fast upwards movement for 7-10 time frames in the middle of the sign (*absent* to *present* to *absent* constraint for y-movement/++). Again, this pattern does a good job in discriminating sign #3

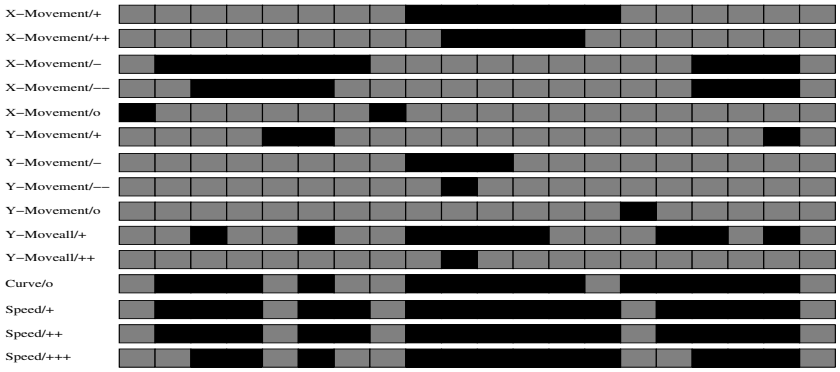

Fig. 13 Pattern found by the beam search with the help of a starting pattern for sign #1.

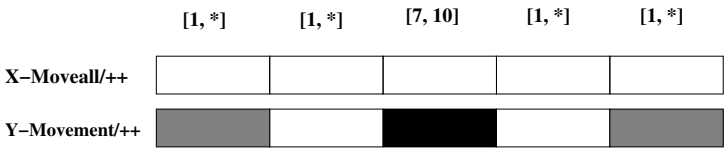

Fig. 14 Pattern found by the beam search without a starting pattern for sign #3.

from all other signs, but it does not help the user to get an impression of sign #3, because it mainly carries information about which predicates are *not allowed* rather than which are *required*.

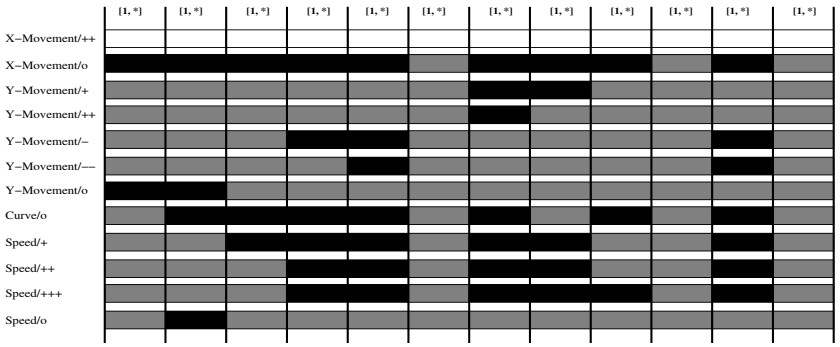

Fig. 15 Pattern found by the beam search with a starting pattern for sign #3.

The pattern which was learned with the help of a starting pattern (Fig. 15) reveals the hand movement pretty well. We recognize that the pattern falls into three parts with *unconstrained* blocks (6th and 10th block), which allow for gaps between the three parts. In the first part the pattern requires no noticeable movement (*present* X- and y-movement/o) at the beginning, and is followed by a downward move (y-

movement $--$), an upward move in the second part, and a downward move in the third part again. No movements to the left or right are allowed as the 'x-movement/o' predicate is *present* most of the time. The *absent* constraint for 'x-movement/++' (top row) was added during the beam search refinement to better discriminate the pattern from all other classes.

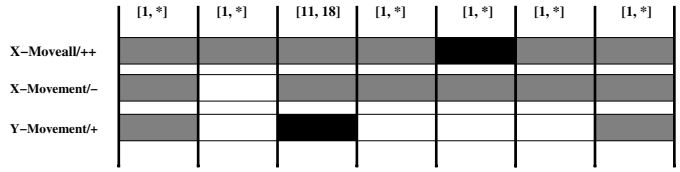

Fig. 16 Pattern found by the beam search without a starting pattern for sign #14.

Sign #14. Finally we inspect the results for sign #14 (cf. Fig. 11). The pattern obtained without using a starting pattern is shown in Fig. 16 and describes an upward move of 11 to 18 time frames and no such upward move before or afterwards. The downward move (which occurs later) is not part of this pattern, because it did not help to discriminate sign #14 from other signs, but it would definitely help the user to interpret the pattern and associate it with Fig. 11.

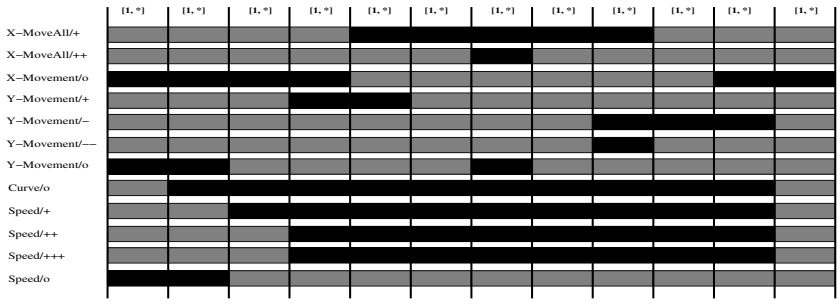

Fig. 17 Pattern found by the beam search with a starting pattern for sign #14.

In Fig. 17 we see the pattern found when using the starting pattern. The pattern is *fully connected*, there is no block with only *unconstrained* predicates. Thus the pattern describes the whole movement without any gaps, which is particularly helpful in reconstructing the hand movement. The pattern requires that there is no movement at the beginning (x/y-movement/o *present*). During blocks 3 and 4 the template describes an upward move (*present* y-movement/+) followed by a combined upward move to the right or left (*present* x-moveall/+ and y-movement/+). In block seven the upward move stops because y-movement/o has to be *present* and the movement in x-direction accelerates (as *present* x-moveall/++ appears). During blocks 8-10, the hand movement in the x-direction decelerates and starts to move

downwards. In the last two blocks the x-movement disappears (as x-movement/o *present* appears) and the downwards move gets slower as well. If we now take into account that during the whole hand movement the speed of the hand is very high and there are no abrupt change of directions because *curve/o* holds, we are able to interpret the pattern as a half circle movement. Another interesting aspect is that the pattern does not state a concrete direction of the x-movement. This is due to the fact that the sign could be drawn from right to left or from left to right.

Some hand movements appear easier to understand by inspecting the plots (e.g. Fig. 11) rather than the obtained patterns (e.g. Fig. 17). But this is only true because the underlying predicates have been extracted from two-dimensional hand movements. In general the data source may consist of more dimensions, mixed binary and numerical sensors, etc., such that no *condensed* representation as in Fig. 11 is possible. We have chosen the libras data set to illustrate that the proposed history templates actually help the user to grasp *what is going on in the data.*

5.2 Effect on Classification Performance

Having discussed the effect on the interpretability, we we now investigate the effect on the classification results. The confusion matrices are shown in Fig. 18 (without starting patterns) and Fig. 19 (with starting patterns).

class	1	2	3	4	5	6	7	8	9	10	11	13	14	15	cannot predict
1	3														1
2		5													1
3			4												
4				3											2
5					2										3
6										1					4
7		1				4		1							
8							3								
9								8							
10										10					3
11											5				3
13												4			4
14													5		
15														5	

Fig. 18 Confusion matrix for the learned patterns without starting pattern with accuracy: 71.765% and error-rate: 28.235%.

class	1	2	3	4	5	6	7	8	9	10	11	13	14	15	cannot predict
1	4														
2		6													
3			4												
4				5											
5					5										
6						4									1
7							4								2
8								3							
9									8						
10										11					2
11											7				1
13												8			
14													5		
15														5	

Fig. 19 Confusion matrix for the learned patterns with starting pattern with accuracy: 92.941% and error-rate: 7.059%.

We can see that accuracy improves by around 21 percent. One reason is the greedy nature of the beam search. During the beam search only constraints that increase the J-measure are added to the pattern, thus refinements which require multiple steps to increase the measure are not found easily due to the myope of the search algorithm. By providing the starting pattern, it is more likely that a critical constraint can be placed right where it is needed, because the basic outline of the pattern is already present right from the beginning. As the intial pattern is constructed such that it matches all histories of one class, the danger of overfitting is not increased despite the high complexity of the pattern.

6 Conclusion

We have investigated the problem of deriving classification rules for temporal or sequential data. The employed classifier operates by successively refining a given pattern to better distinguish between the classes. Instead of learning the patterns for each class *from scratch*, we propose to derive a starting pattern, which consists of those parts that are shared among all instances of the same class (a representative for this class, which has not necessarily any discriminative power). The experimental evaluation has shown that this step not only improves the interpretability of the obtained patterns, but also improves the classification results. Increasing the explanatory power of the patterns [8] and reducing the complexity of searching the starting pattern are topics to be addressed in future work.

References

[1] Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM **26**(11), 832–843 (1983)
[2] Frank, A., Asuncion, A.: UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences (2010)
[3] Höppner, F., Klawonn, F.: Finding informative rules in interval sequences. Intelligent Data Analysis – An International Journal **6**(3), 237–256 (2002)
[4] Höppner, F., Topp, A.: Classification based on the trace of variables over time. In: Proc. Int. Conf. Intelligent Data Engineering and Automated Learning (IDEAL), no. 4881 in LNCS, pp. 739–749. Springer (2007)
[5] Kam, P.S., Fu, A.W.C.: Discovering temporal patterns for interval-based events. In: Proc. of the 2nd Int. Conf. on Data Warehousing and Knowl. Discovery, *LNCS*, vol. 1874, pp. 317–326. Springer (2000)
[6] Mörchen, F.: Time series knowledge mining. Ph.D. thesis, Philipps University Marburg (2006)
[7] Peter, S., Höppner, F.: Finding temporal patterns using constraints on (partial) absence, presence and duration. In: Proc. 14th Int. Conf. on Knowl.-Based and Intel. Inf. & Eng. Sys. (KES). Springer (2010)
[8] Peter, S., Höppner, F., Berthold, M.R.: Pattern graphs: A knowledge-based tool for multivariate temporal pattern retrieval. In: Proc. IEEE Conf. Intelligent Systems. IEEE (2012)
[9] Shahar, Y., Musen, M.A.: RÉSUMÉ: A temporal abstraction system for patient monitoring. Computers and Biomedical Research **26**, 155–273 (1993)
[10] Smyth, P., Goodman, R.M.: An information theoretic approach to rule induction from databases. IEEE Trans. Knowledge Discovery and Engineering **4**(4), 301–316 (1992)
[11] Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. Journal of Computational Biology **1**(4), 337–348 (1994)