Handling Feature Ambiguity in Knowledge Discovery from Time Series*

Frank Höppner

Department of Computer Science
University of Applied Sciences Braunschweig/Wolfenbüttel
Salzdahlumer Strasse 46/48
D-38302 Wolfenbüttel, Germany
frank.hoeppner@ieee.org

Abstract. In knowledge discovery from time series abstractions (like piecewise linear representations) are often preferred over raw data. In most cases it is implicitly assumed that there is a single valid abstraction and that the abstraction method, which is often heuristic in nature, finds this abstraction. We argue that this assumption does not hold in general and that there is need for knowledge discovery methods that pay attention to the ambiguity of features: In a different context, an increasing segment may be considered as (being part of) a decreasing segment. It is not a priori clear which view is correct or meaningful. We show that the relevance of ambiguous features depends on the relevance of the knowledge that can be discovered by using the features. We combine techniques from multiscale signal analysis and interval sequence mining to discover rules about dependencies in multivariate time series.

1 Introduction

Although a complex system is difficult to forecast or model as a whole, such systems (or subsystems thereof) very often cycle through a number of internal states that lead to repetitions of certain patterns in the observed variables. Discovering these patterns may help a human to resolve the underlying causal or temporal relationships and find local prediction rules. Rather than trying to explain the behaviour of the variables *globally*, we therefore seek for *local dependencies* in the data. Now, consider a time series in some neighbourhood around t and assume an increasing trend in this area. This information is probably more reliable than the value at t alone, because a small amount of white noise will not turn an increasing trend into a decreasing trend. There are, however, other effects than white noise, and thus we must be aware that low-frequency disturbances may have caused this increasing trend, but when looking at a coarser scale (zooming out) we have an increasing trend. While our assumption is that there is always a unique sequence of states a system has run through while producing its output, it

* This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) under grant no. Kl 648/1.

is extremely difficult to find the segmentation that corresponds to this sequence, because this means that we have to distinguish all kinds of disturbances to recover the unknown true signal. The difficulty is to distinguish between subtle but important features and uninteresting noise (which is not necessarily Gaussian).

This is illustrated by the time series in Fig. 1. There are two major peaks and right before the decreasing flank of the peaks there is a small peak (and thus a short increasing trend) in both cases. Is that a coincidence? Or is it important? Do we want a heuristic time series conversion procedure decide about that? Better not, because if the peaks are falsely discarded, even the most powerful pattern detection mechanism will not be able to recover their importance. On the other hand, we do not want to consider every noisy data point separately, since this would increase the computational cost of the pattern discovery process significantly. The importance of such a small peak can only be shown a posteriori by some interesting pattern that uses the peak. Surprisingly, most of the work in KDD from time series, e.g. [3, 8, 4], use a *single* abstraction of the time series and thus the decision whether such patterns will be detected or not is implicitly handed over to the abstraction method. In this paper we provide an extension of [4] that takes ambiguity into account.

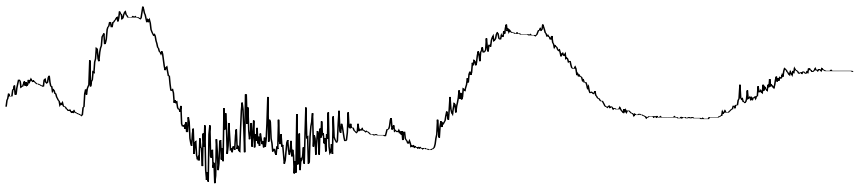

Fig. 1. A noisy time series.

2 An Overview of the Approach

In this section we outline our approach to knowledge discovery from time series and the contribution of this paper. At the beginning, the time series are converted into a higher-level representation, that is, a labeled interval sequence with labels addressing aspects like slope or curvature. The sequence consists of triples (b, f, s) denoting an interval $[b, f]$ in time for which the description $s \in \mathcal{S}$ holds ($\mathcal{S}$ denotes the set of all possible trends or properties that we distinguish).

We use Allen's interval relationships [1] to describe the relationship between intervals, for example, we say "A meets B" if interval A terminates at the same point in time at which B starts. In the following we denote the set of interval relations as shown in Fig. 2 by $\mathcal{I}$. A temporal pattern of size k is defined by a pair $P = (s, R)$, where $s : \{1, .., k\} \rightarrow \mathcal{S}$ denotes the label of interval $\#i$, and $R \in \mathcal{I}^{k \times k}$ denotes the relationship between $[b_i, f_i]$ and $[b_j, f_j]$. By $\dim(P)$ we denote the number of intervals k of the pattern P, we also say that P is a k-pattern. The whole sequence is sorted lexicographically [7] and intervals are numbered according to their position in the sorted list. Finding an embedding π

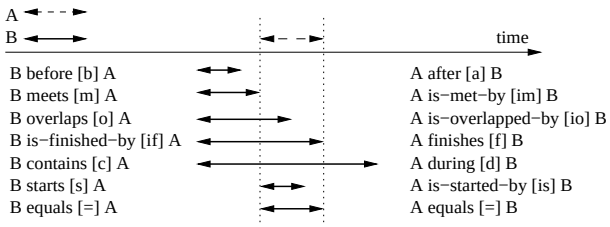

Fig. 2. Allen's interval relationships.

that maps interval $\#i$ of a pattern P to interval $\#j$ in the sequence ($\pi(i) = j$) such that all interval relationships hold corresponds to searching an *instance* of a temporal pattern in the interval sequence.

To be considered interesting, a temporal pattern has to be limited in its extension. We therefore choose a maximum duration t_{max}, which serves as the width of a sliding window which is moved along the sequences. We consider only those pattern instances that can be observed within this window. We define the total time in which the pattern can be observed within the sliding window as the support supp(P) of the pattern P. We restrict our attention to so-called *frequent patterns* that have a support above a certain threshold. For the efficient support estimation of temporal patterns (see [4, 7] for the details), we make use of a simple pruning technique: once we have observed an instance of a temporal pattern, we can stop any further checking until this instance disappears from the window. In order not to miss any instances, we must then be sure that we always find the *earliest* instance of a pattern, otherwise we loose the soundness: Suppose there are two instances at t_0 and t_1. If we detect the instance at t_1 and postpone any further checking until this instance disappears it may happen that the instance at t_0 has also disappeared before we perform the next check and thus we loose the support in $[t_0, t_1]$. To ensure correctness we have shown

Theorem 1. *Given two instances ϕ and ψ of the same pattern P, then $\pi = \min(\phi, \psi)$ $(i \mapsto \min(\phi(i), \psi(i)))$ is also an instance of P and $b_\pi \leq \min(b_\phi, b_\psi)$ holds. (For any instance ϑ, b_ϑ is the point in time when we start to observe ϑ.)*

Thus, finding the *earliest* instance of a pattern corresponds to finding the first mapping π (in a lexicographical ordering of vectors $(\pi(1), \pi(2), ..., \pi(\dim(P)))$) and we make use of this fact in the subpattern test in [7].

There is one important point about our interval sequence that has not been mentioned so far. While we did not require that one interval has ended before another interval starts (which enables us to mix up several sequences into a single one), we required that every labeled interval (b_i, f_i, s) is *maximal* in the sense, that there is no (b_j, f_j, s) in the sequence such that $[b_i, f_i]$ and $[b_j, f_j]$ intersect:

$$\forall (b_i, f_i, s_i), (b_j, f_j, s_j), i < j : f_i \geq b_j \Rightarrow s_i \neq s_j \qquad (1)$$

The idea is that whenever (1) is violated, we can merge both intervals and replace them by their union $(\min(b_i, b_j), \max(f_i, f_j), s)$. However, in Fig. 1, we have discussed the question whether the decreasing flank of the major peaks shall

be considered as a single decreasing flank within some interval $[a, d]$ or a sequence of increasing, decreasing, and increasing intervals within $[a, b]$, $[b, c]$, and $[c, d]$ $(a < b < c < d)$. It is (1) that prevents us from composing our interval sequence out of all these intervals, because (1) does not allow $[a, d]$ and $[c, d]$ to have the same label. Unfortunately, we have used (1) in the proof of Theorem 1 – thus we loose the soundness of our approach if (1) is abandoned. In the remainder of this paper we discuss how to recover the soundness and efficiency of the approach when eliminating (1) and considering ambiguous data abstractions.

3 Multiscale Feature Extraction

By "feature" we refer to the labels of the interval sequence, such as increasing or decreasing trend. How can we create an ambiguous abstraction of a time series that uses such features? The transition between increasing and decreasing trend is indicated by a zero-crossing in the first derivative, however, noise introduces many zero-crossing and the resulting representation would not correspond to the human perception of the profile. By analysing the signal in multiple scales, that is different degrees of smoothing, and comparing the abstractions against each other, a concise description of the time series (cf. Fig. 3) by means of an *interval tree of scale* (cf. Fig. 4) can be developed [9]. Each of the rectangles in the tree defines a qualitative feature (here either increasing or decreasing, indicated by '+' and '-'). The horizontal extent of the rectangle defines the valid time interval for this feature and the vertical extent defines the valid range in scale. A large vertical extent indicates that the feature is perceived over a broad range of smoothing filters and thus represents a robust feature, whereas rectangles with small vertical extent correspond to noisy features. Due to lack of space, we neither explain the method in detail nor justify our choice, but refer the reader to [9, 2] and [5], resp.

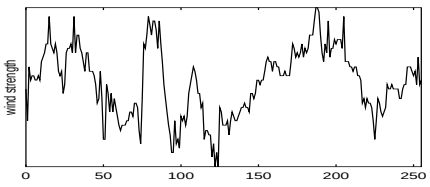

Fig. 3. Wind strength over 10 days.

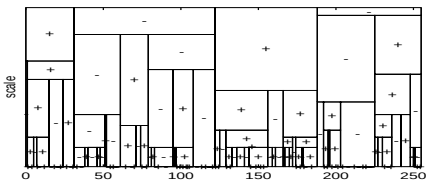

Fig. 4. Interval Tree of Scale.

4 Handling Ambiguity in the Discovery Process

We can circumvent the elimination of (1) by renaming the labels of all intervals such that they reflect the scale from which they have been extracted. A label $s \in \mathcal{S}$ is no longer used alone but only in combination with a scale, like $s - 4$ indicating that the description s holds for scale 4. While "A starts A" is not

allowed, "$A-2$ starts $A-4$" does not violate (1). However, this approach increases the data volume (if an interval labeled A survives over scales 1 to 5, we have to add it 5 times to the interval sequences with label $A-1$ to $A-5$) and does not match identical patterns on different scales ($A-4$ does not match $A-3$). Therefore, we prefer not to incorporate the scale in the label. Then it is sufficient to consider only one interval per rectangle in the interval tree of scale. To follow this approach we have to find a new subpattern check. The naive approach is to enumerate all occurrences of a temporal pattern and then yield the one that can be observed first, however, this appears to be ineffective due to the potential combinatorial explosion of possible embeddings.

Let us recall how the visibility of a temporal pattern is determined [7]: A pattern becomes visible if the interval bound next to the rightmost interval bound coincides with the right bound of the sliding window and invisible if the interval bound next to the leftmost interval bound coincides with the left bound of the sliding window. In previous work we have calculated the observation interval *a posteriori* given a detected instance π of a pattern P. But since *all* instances of P have the same qualitative structure and thus the same order of the interval bounds, it is possible to compute in advance which are the two relevant bounds[1]. Having identified these two bounds and the corresponding intervals $\#i$ and $\#j$ of P, we may calculate the observation interval of a potential instance *a priori*: we only have to know the two intervals that we intend to use for $\#i$ and $\#j$ rather than the embedding π of the whole pattern.

Instead of a brute force enumeration of all possible embeddings in a naive approach, we can now organize the search for valid embeddings in such a way that the first match yields the earliest instance: For every pair of labels (s,t) and temporal relationship r, we maintain a list of matching interval pairs in the sliding window. These lists are kept up to date incrementally whenever the content of the sliding window changes. Thus, for any pattern P we easily find all possible pairs of intervals that determine the observation interval. Sorting these pairs by the beginning of the expected observation interval and using this list to control the search for possible embeddings guarantees that we will find the earliest instance first. Besides that, we use the lists of interval-pairs to influence the backtracking depth of the subpattern check routine. For the remaining intervals of the pattern, we look for the interval-pair with the smallest number of occurrences in the sliding window. The fewer possibilities the lower the probability of intensive backtracking.

5 Loosely Connected Patterns

With the elimination of (1) more temporal patterns are allowed and thus we have implicitly increased the size of the pattern space. In this section we propose to remove some other patterns from the pattern space to compensate this increase.

[1] The algorithm is somewhat troublesome but nevertheless straightforward. No sketch is given due to lack of space, contact author for detailed report.

In most other approaches where (local) time series similarity is decided with the help of a sliding window (e.g. dynamic time warping), to be considered as similar the *complete* content of one window position has to match the *complete* content of another window position. In our approach, the window is not that restrictive, it mainly provides an upper bound for the temporal extent of patterns, while the content of the sliding window at different positions has to match only *partially* – depending on the pattern we are currently looking for. Therefore, it makes sense to experiment with larger sliding windows.

Suppose that we have an interesting pattern P. The probability of observing pattern "P overlaps A" or "P meets C" does not increase significantly with the window width: such patterns can already be observed with smaller window width and the number of occurrences increases only slightly as the window width is increased. However, patterns "P before A" or "A before P" become more frequent, simply because it is more and more likely that we can observe some more A instances as the width of the sliding window increases – far apart and without any relationship to P. Thus, a rule $P \rightarrow A$ does not necessarily indicate a relationship between P and A, it may have arisen from a large window width.

We call a temporal pattern P *loosely connected* (LC-pattern), if the (undirected) graph $G = (V, E)$ is connected, where the set of vertices is given by $V = \{1, .., \dim(P)\}$ and the set of edges is given by $E = \{(i, j) \mid R[i, j] \in \mathcal{I}_L\}$ with $\mathcal{I}_L = \mathcal{I} \backslash \{\text{after, before}\}$. Note that this definition does not exclude after and before relationships in P in general. Intuitively, a pattern is loosely connected if we can draw a vertical line through the pattern without intersecting intervals and thereby separating the intervals into two groups. We want to restrict the pattern enumeration process to LC-patterns. To do this, we have to make sure that during candidate generation all and only LC-patterns are generated.

For the pruning of candidate patterns in association mining we use the fact that any subpattern of a frequent pattern is itself a frequent pattern. This is no longer true for loosely connected patterns (consider "A overlaps B, B overlaps C" and the removal of B). However, we have the following observation: Given a (loosely) connected pattern P with $\dim(P) \geq 2$. Then one can find at least two different LC-subpatterns Q and R with $\dim(Q) = \dim(R) = \dim(P) - 1$. Among the $(k-1)$-subpatterns there are at least 2 subpatterns which are loosely connected. From those we can construct a set of candidate patterns similar as it has been done before [4]. However, with respect to pruning efficiency we expect the new candidate generation algorithm to generate more candidates than before because we have fewer LC-subpatterns that can be used for pruning. (Algorithms are omitted due to lack of space, request detailed report from author.)

6 Evaluation

In our experiments, the number of intervals in a multiscale description was 2–3 times the number of intervals in a single scale description. Increasing the number of intervals while keeping the set of labels constant may cause a dramatic increase in the number of frequent patterns (much more relationships contains,

starts, finishes, etc.). But on the other hand, only a small percentage of the frequent patterns are loosely connected patterns (percentage decreases drastically with increasing window width), therefore the reduced pruning efficiency in case of loosely connected patterns is compensated by the savings during support estimation. The pruning techniques are less efficient, the ratio of frequent patterns among candidate patterns decreases. This is due to the fact that we have fewer loosely connected subpatterns that can be used for pruning. In our application to weather data we obtained much better supported patterns than without considering ambiguous abstractions, which we take as an indication that the previously used abstraction method did not always provide the best segmentation possible.

For illustration purposes we demonstrate the method using an artificial example, which has been generated by concatenating noisy squared and unsquared sine waves ($\sin(\frac{2\pi}{l}t)$, $t \in \{0,..,l\}$ with l varying randomly within 200 and 300). For some $\tau \in \{\frac{1}{16}, \frac{7}{16}, \frac{9}{16}, \frac{15}{16}\}$ we randomly added a Gaussian bump ($\exp(-(t-\tau*l)^2/100)/h$ with h varying randomly within 2 and 3). The sine wave is squared if a Gaussian bump appears at $\tau = \frac{1}{16}$. Figure 5 shows an excerpt of this time series. The difficulty is to distinguish the important bump ($t = \frac{1}{16}$) from those that have no special meaning ($\tau \in \{\frac{7}{16}, \frac{9}{16}, \frac{15}{16}\}$). It is not possible to generate a single abstraction of the time series that contains only the important bumps, since all bumps have the same characteristics. Their importance can only be revealed by means of the rules that can be discovered by using them.

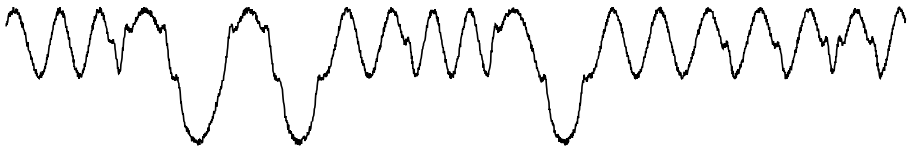

Fig. 5. The "sine wave with bumps" example.

We ran the discovery process (window width 200, $\text{supp}_{min} = 5\%$) with increasing and decreasing segments only. Then we performed specialization of the best discovered rules [6] to get some evidence for useful thresholds on segment lengths. From the histogram of thresholds on interval lengths we identified two major peaks at segment length ≈ 50 and ≈ 90. Therefore the process was restarted with refined labels denoting the length of the segment (label prefix "`short`" if ≤ 50, label prefix "`long`" if ≥ 90). We specialized the discovered rules that contained a "`long-dec`" label and the best rule with "`long-dec`" in the conclusion was

> If | `short-dec` | `inc` | has been observed with a gradient between 0.006-0.009 and a length ≥ 24 for the `inc` segment, then | `short-dec` | `inc` | `long-dec` | will be observed.

The rule has a support of 10% and confidence of nearly 100%. The premise contains a short decreasing segment (from a Gaussian bump) that meets an increasing segment of the remaining sine wave. A long decreasing segment (≥ 90) is only obtained for an unsquared sine wave, therefore this rule recognizes the relationship between the first bump and the squaredness of the sine curve.

The squared and non-squared sine waves distinguish slightly in their derivative, and the additional quantitative constraint on the gradient obtained from rule specialization [6] focuses on the non-squared sine wave.

7 Conclusions

In knowledge discovery from time series one has to carefully balance the features of the used time series similarity measure and the computational cost of the discovery process. The proposed approach via labeled interval sequences supports partial similarity of time series segments and can handle gaps, translation and dilatation to some extent. Furthermore, it can be applied not only to univariate but also to multivariate time series, and the representation is close to the human perception of patterns in time series. Ambiguity in time series perception is an important issue and we have shown in this paper that certain relationships in the data cannot be revealed if only a single abstraction is used. We have extended our approach by the ability of taking ambiguity in time series perception into account, which is a feature that is absent in most competitive approaches.

References

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Comm. ACM*, 26(11):832–843, 1983.

[2] B. R. Bakshi and G. Stephanopoulos. Reasoning in time: Modelling, analysis, and pattern recognition of temporal process trends. In *Advances in Chemical Engineering*, volume 22, pages 485–548. Academic Press, Inc., 1995.

[3] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *Proc. of the 4th Int. Conf. on Knowl. Discovery and Data Mining*, pages 16–22. AAAI Press, 1998.

[4] F. Höppner. Discovery of temporal patterns – learning rules about the qualitative behaviour of time series. In *Proc. of the 5th Europ. Conf. on Principles of Data Mining and Knowl. Discovery*, number 2168 in LNAI, pages 192–203, Freiburg, Germany, Sept. 2001. Springer.

[5] F. Höppner. Time series abstraction methods – a survey. In K. Morik, editor, *GI Workshop on Knowl. Discovery in Databases*, LNI, Dortmund, Germany, Sept. 2002.

[6] F. Höppner and F. Klawonn. Finding informative rules in interval sequences. In *Proc. of the 4th Int. Symp. on Intelligent Data Analysis*, volume 2189 of *LNCS*, pages 123–132, Lissabon, Portugal, Sept. 2001. Springer.

[7] F. Höppner and F. Klawonn. Learning rules about the development of variables over time. In C. T. Leondes, editor, *Intelligent Systems: Technology and Applications*, volume IV, chapter 9, pages 201–228. CRC Press, 2002.

[8] E. J. Keogh and P. Smyth. A probabilistic approach to fast pattern matching in time series databases. In *Proc. of the 3rd Int. Conf. on Knowl. Discovery and Data Mining*, pages 20–24, 1997.

[9] A. P. Witkin. Scale space filtering. In *Proc. of the 8th Int. Joint Conf. on Artifial Intelligence*, pages 1019–1022, Karlsruhe, Germany, 1983.