Motif-based Classification of Time Series with Bayesian Networks and SVMs

Krisztian Buza and Lars Schmidt-Thieme

Information Systems and Machine Learning Lab, University of Hildesheim, Marienburger Platz 22, 31141 Hildesheim, Germany
{buza,schmidt-thieme}@ismll.de

Summary. Classification of time series is an important task with many challenging applications like brain wave (EEG) analysis, signature verification or speech recognition. In this paper we show how characteristic local patterns (motifs) can improve the classification accuracy. We introduce a new motif class, generalized semi-continuous motifs. To allow flexibility and noise robustness, these motifs may include gaps of various lengths, generic and more specific wildcards. We propose an efficient algorithm for mining generalized sequential motifs. In experiments on real medical data, we show how generalized semi-continuous motifs improve the accuracy of SVMs and Bayesian Networks for time series classificiation.

Key words: Time Series, Motifs, Bayesian Networks, SVM

1 Introduction

Many phenomena are quantitatively changing continuously in time, like blood pressure or body temperature of a person, exchange rates of currencies, speed of a car, etc. Making observations regularly results in a sequence of measured values (usually real numbers). We call such a sequence a *time series.*

We illustrate classification of time series on an example. Suppose we are trading with currencies, and we know the past exchange rates for some currencies.

We are interested in the exchange rates in the future. Now we can group currencies into three classes based on how the exchange rate changes in the near future (next month): *i)* the exchange rate *increases* at least by 5 %, *ii)* the exchange rate *decreases* at least by 5 %, *iii)* the exchange rate does *not change* significantly. Based on the times series representing exchange rates in the past, we would like to predict exchange rates for the next month, i.e. we want to classify time series of currencies in one of the previously defined classes.

In general, classification of time series is an important task related to many challenging practical problems like indexing of handwritten documents

[24, 19], speech recognition [27], signature verification [11], analysis of brain wave (EEG) signals [20] or medical diagnosis [16].

In this paper we focus on the classification of time series based on recurrent patterns, called motifs. We show that time series classification models based on SVMs and Bayesian Networks can be improved using motifs. As the main contribution we introduce a new motif class: a common generalization of continuous and non-continuous sequential mofits, called generalized semi-continuous motifs. We propose an efficient algorithm for mining generalized sequential motifs. We compare generalized semi-continuous motifs to existing motif classes, and show that generalized semi-continuous motifs outperform other classes of motifs in the time series classification task.

2 Related Work

Motif Discovery. The *task of motif (or pattern) discovery* in time series is understood in slightly different ways in the literature. Yankov at al. [31] and Patel at al. [21] define the task of motif discovery regarding one "long" time series: the target of their work is to identify recurrent patterns, i.e. approximately repeated parts of the given time series. In contrast, Futschik and Carlisle [9] are concerned with a set of time series. They apply global patterns: they cluster times series, and calculate the "compromise" time series for each cluster. Such a "compromise" time series is regarded as a representative pattern of the time series in the cluster. Jensen at al. [13] and Ferreira at al. [7] also use clustering, however in a more local fashion: they do not cluster the whole sequences, but subsequences of them.

Predefining a (minimal) length L for motifs, scanning the database, and enumerating (almost) all the subsequences of the given lenght L is common in the biological domain [31, 7, 13, 21]. However this may not be efficient enough, especially if complex motifs with gaps and/or taxonomical wildcards are to be discovered (see Fig. 1). For noise-robust motif detection (without wildcards) Buhler and Tompa use random projections [5]. A more sophisticated solution is based on the *antimonotonicity* originally observed in the frequent itemset mining community [1], namely, that subpatterns of a frequent pattern are also frequent. State of the art frequent sequence mining algorithms are based on antimonotonicity [10, 2]. It suggests, roughly speaking, that one discovers "short" motifs first and somehow "grows" them step by step together to longer ones. Approaches based on the antimonotonicity avoid processing of many redundant (i.e. non-motif) subsequences. They have intensively been researched resulting in highly efficient implementations [3, 4]. For databases of different character there are special algorithms: for example in case if many short motifs are expected, they can efficiently be discovered ("grown together") in a breadth-first search manner like in [2], if long motifs are expected, approaches of a depth-first search fashion are preferable, like PrefixSpan [22].

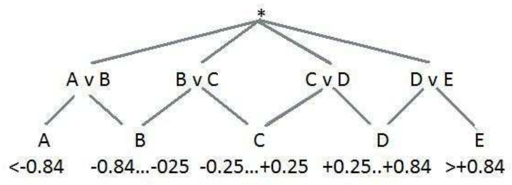

Fig. 1. Specific and generic wildcards build a taxonomy of symbols

In this work, we mean by motifs approximately repeated *local patterns*. We propose an approach based on *pattern mining* techniques in order to discover motifs regarding a *set of time series*.

As usual in time series motif detection [16, 17, 21], as preprocessing step, we turn time series into a sequence of discrete symbols using Symbolic Aggregate Approximation [18]. In our experiments we use 10 (exp. 1) and 7 (exp. 2) different symbols, we aggregate on an interval of length 4.

According to this representation, different classes of motifs can be defined regarding generality and character. Regarding generality, we distinguish *i) flat patterns* (without wildcards), and *ii)* patterns with taxonomical wildcards (see Fig. 1), called *generalized* patterns. Regarding character we distinguish between *i)* Set Motifs (the order of symbols is omitted), *ii)* Sequential Motifs (continuous and non-continuous ones), and *iii)* Semi-continuous Sequential Motifs, which is a common generalization of continuous and non-continuous motifs: in semi-continuous motifs maximal n gaps of each of a maximal length d is allowed. (For $d = n = 0$ semi-continuous motifs are ident to continuous motifs; for $d = n = \infty$ they are the same as non-continuous motifs.) Table 1 provides a systematic overview of selected works on pattern mining.

For the task of discovering flat patterns optimization techniques (recursive counting and recursion pruning) were introduced in [3, 4]. To the best of our knowledge they have not been generalized for patterns with taxonomic wildcards yet. Ferreira and Azevedo [8] have already allowed gaps in motifs, however without taxonomical wildcards. They discovered motifs by enumerating all subsequences (of a given length), they have not used an algorithm exploiting pattern mining techniques.

Motif-based Classification. Motifs have been used for sequence classification in biological domain [6, 15, 8]. This is usually done in two steps: *i)* first motifs are extracted, then *ii)* each time series is represented as an attribute vector using motifs so that a classifier like SVM [15], Naive Bayes [8], Decision Tree [6], etc. can be applied. Some possible ways of construction of attributes are: *i)* there is a binary attribute for each motif, which indicates if the motif is contained in the time series or not [16, 6, 15] (see Fig. 2), *ii)* aggregating attributes may indicate the total count and/or average length of motifs occurring in a time series[8].

Time Series Classification. As a baseline in our experiments we have chosen the Nearest Neighbour approach with DTW as distance measure. DTW is

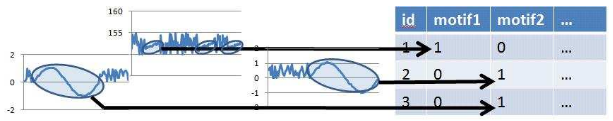

Fig. 2. Representation of time series as an attribute vector using motif features.

Table 1. Systematic Overview of Selected Pattern Mining Related Work

	Flat Patterns	With taxonomy
Set Motifs (item sets)	[1, 3, 4]	[12, 23, 29, 30]
Sequential Motifs (continuous, non-cont.)	[2]	[10, 28]
Semi-continuous Sequential Motifs	[8]	this work

basically an edit distance, allowing stretching of time series. It was introduced in [27]. There are recent works on DTW, to make it more accurate [24], and speed up to data mining applications [14]. It has recently been studied from a theoretical-empirical point of view [25]. Furthermore there is some recent work that suggests that DTW is the best solution for some time series classification problems [26]. Thus, DTW is state of the art.

3 Discovery of Generalized Semi-Continuous Motifs

To solve the time series classification task, first we search for motifs in the time series. In this section we describe our motif discovery approach in detail. We suppose that time series are converted to a sequence of discrete symbols. Motif discovery means finding frequent subsequences in the dataset consisting of time series sequences.

Definitions. Given a database of time series D, a set of symbols Σ and a taxonomic relation T_Σ over Σ, the maximal number of gaps n, the maximal length of gaps d, and a minimum support threshold s. A ***symbol*** $x \in \Sigma$ ***matches*** an other symbol $y \in \Sigma$ if either $x = y$ or y is a descendant of x according to T_Σ, x is matching symbol, y is called matched symbol. A ***sequence of symbols*** m ***semi-continuously matches*** a times series $t \in D$, if all symbols of m match at least one symbol in t so that *i)* the order of matched symbols are the same as the order of matching symbols, and *ii)* the matched symbols in t build a basically continuous sequence but maximal n gaps with maximal length d are allowed. A sequence of symbols m is called ***semi-continuous motif***, if it matches at least s time series in T. The number of matched time series is called ***support*** of m.

Checking the support of all possible sequences whether they are motifs or not, is not feasible as it features an inherent unaffordable high computational cost due to the large number of possible sequences. Thus, we need to prune the search space in order to reduce the number of sequences to be checked and need an efficient implementation for checking supports. We adapt and combine constraints in [1] and in [29, 30] for semi-continuous generalized motifs, and we extend optimization ideas in [3, 4] for generalized semi-continuous motifs.

The basic intuition behind the constraints on generalized semi-continuous motifs is the use of the antimonotonous property of the support function: *i)* if a sequence p includes another sequence p', the support of p is less than or equal to the support of p', *ii)* if a sequence p is less general than some other sequence p', the support of p is less than or equal to the support of p'.

These conditions hold as the number of time series matching p can not be higher than the number of time series matching p', as every time series t matching p also matches p'. The following formalization of these constraints define how inclusion and generalization are exactly meant. Both constraints are consequences of the definition of the support (and semi-continuous matching). Let p be a sequence over Σ (eventually including taxonomic wildcards): $p = (w_1, w_2, \dots, w_{k-1}, w_k)$, each $w_i \in \Sigma$, $0 < i \leq k$.

Constraint 1. Let p' be subsequence of p: $p' = (w_j, w_{j+1}, w_{j+2}, \dots w_{j+l})$, $0 < j \leq j + l \leq k$. In this case $\mathit{support}(p) \leq \mathit{support}(p')$.

Constraint 2. Denote the transitive closure of the taxonomic relation T_Σ with T_Σ^* (i.e. $(x, y) \in T_\Sigma^*$ means x is a descendant of y in the taxonomy). Suppose p' is a more general sequence than p: $p' = (w'_1, w'_2, \dots, w'_{k-1}, w'_k)$ with $\forall i : (w_i, w'_i) \in T_\Sigma^*, 0 < i \leq k$. In this case $\mathit{support}(p) \leq \mathit{support}(p')$.

These constraints suggest to check the shorter and more general sequences first for being motifs or not. For example, if we are given the taxonomy in Fig. 3, pattern (G, H) would be checked before (G, w) and (G, H, H).

For motif mining we use a significantly extended version of the algorithm Apriori [1]. The Apriori algorithm essentially iterates over three steps: *i) Candidate generation:* Based on motifs found in the previous iterations, some new sequences will be chosen in order to be checked for support. They are *candidates. ii) Support counting:* The support of each candidate will be determined. *iii) Filter infrequent candidates:* The candidates with less support than the given threshold s are deleted. The other ones are motifs.

The computational cost of Apriori highly depends on the applied data structure. Tries have been shown to be efficient (see [3]). In a trie a path from the root to a node encodes a sequence. A simplified view of our datastructure is shown in Fig. 3. For example the path $(\mathit{root}, G, J, H, w)$ encodes the sequence (J, w). There are two different types of edges in this path: there are *taxonomical* and *sequential* edges (straight lines and straight arrows). Sequential edges are (root, G) and (J, H). Each sequential edge denotes a new symbol of the sequence. The taxonomical edges in the path specialize the symbols. In this path G was specialized to J, and H was specialized to w.

There are also "cross-pointers" in the data structure, pointing from a sequence prefix $(w_1, w_2, \dots, w_k)$ to the sequence prefix $(w_2, w_3, \dots, w_k)$. To keep the example simple, only one cross pointer is depicted (dashed arrow). These "cross-pointers" allow quick candidate generation.

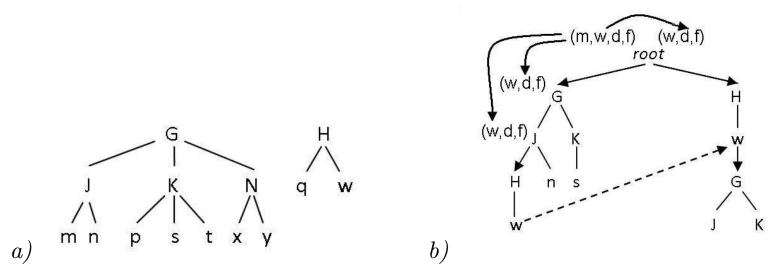

Fig. 3. *a)* An example taxonomy with two roots. *b)* A simplified overview of the data-structure for counting candidates and storing motifs. Straight arrows denote sequential children, lines denote taxonomic children. One of the cross-pointers is depicted with dotted arrow. Curved arrows show the recursion steps in the double recursive search scheme.

Candidate generation: Let c_r denote the count of roots in the taxonomy T_Σ. At the beginning of the first iteration, there are c_r candidates, one for each root. These are the most general and shortest sequences (they consist of one item). After counting the support, the candidates for the next iteration are always calculated based on the motifs already found. As application of Constraint 2, a motif p of length 1 may have any concretizing extensions p' conform to the given taxonomy. For longer motifs: knowing that the sequence $p = (w_1, w_2, \ldots, w_{k-1}, w_k)$ is a motif, its concretizing extension $p' = (w_1, w_2, \ldots, w_{k-1}, w'_k)$, $(w'_k, w_k) \in T^*_\Sigma$ may only be a motif if $p_i = (w_2, \ldots, w_{k-1}, w'_k)$ is a motif.
As application of Constraint 1, a motif p of length 1 may sequentially be extended by any other motif p_{ii} of length 1. Thus the new candidate p'' is generated. For longer motifs: knowing that the sequence $p = (w_1, w_2, \ldots, w_{k-1}, w_k)$ is motif, its sequential extension $p'' = (w_1, w_2, \ldots, w_k, w_{k+1})$, may only be motif if $p_{ii} = (w_2, \ldots, w_{k-1}, w_k, w_{k+1})$ is motif. When generating candidates, we always apply all possible sequential and concretizing extensions. This has the advantage that we always know in advance whether the sequences p_i and p_{ii} are motifs. Furthermore, the cross-pointers have to be updated as well.
Support count: The dataset of time series is processed sequentially, one sequence at a time. For each node the trie contains a counter. For each time series, the counters of the matched candidates are incremented.

Matched candidates can be found efficiently using a double recursive search scheme. The doubly recursive search scheme is shown in Fig. 3 and Fig. 4.

When processing a time series, the function for counting is first invoked for the root. (The current node is the root.) Then, this function is recursively invoked with the tail of the current time series for *i)* such a sequential child of the current node which match the first item of the time series, and for *ii)* all matching taxonomic descendants of the matching sequential child *iii)* for the current node as well. Note that this step is a generalization of the corresponding step in [3].

```
support_count() {
  for each time series (denoted by s) of input
      support_count_1(candidates.root, s, max_dist);
      if (s did not support any node) delete s from input;
}
support_count_1(TrieNode n, TimeSeries s, int allowed_dist_to_next_match) {
   item first_symbol = s.first();
   TailTimeSeries tail_sequece = s.tailTimeSeries();
   TrieNode n1 = sequential child node of n such that
                the incoming edge to n1 matches first_symbol;
   if (exists n1) && (exists a candidate reachable over n1)
(*)  N = set of n1 and all of the taxonomic descendants of n1 matched by first_symbol
     for each node (denoted by n0) of N
       if (n0 candidate) && (n0 has not been supported by this input sequence before)
          n0.incrementSupport();
       support_count_1(n0,tail_sequence, max_dist);
(*)if check(allowed_dist_to_next_matching)
     support_count_1(n, tail_sequence,new_allowed_dist_to_next_matching);
}
```

Fig. 4. Pseudo-code of one of the main steps of our algorithm: support counting, lines marked with (*) contain generalization compared to the case of flat motifs

In contrast to [3], in our case we need some additional administration. During this double recursive search, we have to take into account (and eventually not invoke some of the recursion steps because of) *i)* the number of "gaps" that the (d, n) semi-continuous candidate has already had in the input sequence till the current position, and *ii)* (if there is currently a "gap" in the matched time series) the length of the current "gap" till the current position. We also have to take care of not incrementing the counter of a node twice while processing a time series. (Note that it is possible to arrive several times at the same node as a candidate may be matched by several parts of a time series.) To further increase the efficiency, we use a pruning technique similar to the one described in [4], i.e. pruning those subtrees which do not contain candidates.

4 Experimental Evaluation

Data set. The data used in our experiments was collected at the Fresenius Clinics. It contains recordings of dialysis sessions for 725 patients. The patients have to consult the doctor for treatment regularly, some data (like blood pressure, body temperature...) is recorded every time, this leads to a sequence of observations. We have about 40 time series per patients. Some pieces of master data of the patients (like age, sex, body mass index,...) are also available. There are two groups of patients: "normal" (53%) and "risky" (47%). We use the same dataset as in [16, 17]. We refer to [17] for a more detailed description.
Experimental Settings, Motif Selection. We discover motifs on different time series separately (i.e. separately on the time series of blood pressure, body temperature, ...). Minimum support threshold was set to 0.06 (exp. 1)

and 0.05 (exp 2). Similar to [16] we select the best predictive motifs for each class: we choose motifs that predict the "normal" class with a probability of 90% (exp. 1c), 85% (exp. 1a,b), 80% (exp. 2) and the motifs that predict the "risky" class with a probability of 85% (exp. 1c), 80% (exp. 1a,b), 75% (exp. 2). Furthermore we only select motifs that are statistically significant for a class (χ^2 test, $\alpha = 0.05$) and limit the total number of apriori-iterations to 10 (exp. 1) and 20 (exp. 2) in order to get *local* motifs. In exp. 1 we limit the minimum length of motifs as well to 3 symbols. Among the motifs fulfilling these criteria, in exp. 1. we only select 5 motifs for each of the 40 different kind of time series the following way: first we select the motif m_0, which is supported by the most of the time series. Then we perform 4 iterations and always select the motif m_k ($k = 1, 2, 3, 4$), which is supported by the most of such time series, which do *not* support $m_0, \dots, m_{k-1}$.

As central classifier we use the WEKA-implementation of SVM-s (with RBF kernel) and Bayesian Networks. The parameters of the SVM-s (complexity constant and exponent) are learned in $2^{-10} \dots 2^3$ and $2^{-10} \dots 2^{11}$ using a hold-out subset of training data in 5-fold-crossvalidation protocol. We perform 10-fold-crossvalidation (i.e. the full dataset is split into test and train set 10 times). Motifs are discovered on the training set. Then time series of the test set are checked whether they contain the motifs discovered on the training set. Note that our experimental protocol differs from the one used in [16], thus our results are not directly comparable.

To calculate the baseline, in exp. 1. we use time series aggregates data (min., max., avg.) as input of the central classifier. We extend it with CPMs attributes (CPM = Count of Predictive Motifs for each class). In exp. 2. we use both master data and time series aggregates (baseline). We extend it with features indicating the containment of each motif like in Fig. 2 and with CPMs attributes as well. We also compare to an SVM classifier integrating kNN-DTW predictions and master data (MD).

Results. In exp. 1. we compare different subclasses of semi-continuous generalized motifs [*i)* continuous, *ii)* semi-continuous with max. 2 gaps of max. length 1, *iii)* motifs with taxonomical wildcards using a simple taxonomy like in Fig. 1 but without $*$]. The results (Table 2.) show that both gaps and taxonomical wildcards are beneficial for classification accuracy. Exp. 2. (Table 3.) shows, that motifs are beneficial in a realistic scenario (where master data is also available) as well.

Table 2. Impact of different generalized semi-continuous motif subclasses (exp. 1)

	without motifs	**a)** continuous motifs (counts)	**b)** (1,2) semi-cont. motifs (counts)	**c)** motifs with tax. wildchards (counts)
SVM	66,52	65,07	68,24	68,13
SVM (logistic)	65,97	66,28	67,00	68,39
Bayesian Network	70,11	70,51	71,04	69,37

Table 3. Impact of motifs if master data (MD) is available (exp. 2)

	Baseline (without motifs)	motifs (counts)	motifs (counts + indicators)
SVM	72,40	72,12	75,61
SVM (logistic)	72,38	73,35	76,43
Bayesian Network	73,84	74,76	74,76
SVM on kNN-DTW + MD	73,7 [17]		

5 Conclusion

We introduced a new class of motifs, generalized semi-continuous motifs. We proposed an efficient algorithm to discover them, and we showed that these motifs improve the accuracy of time series classification. As future work we would like to compare different subclasses of generalized semi-continuous motifs in more detail and deal with parameter learning.

References

1. R. Agrawal, R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. *20th Int'l Conf. on Very Large Data Bases*, pages 487–499, 1994.
2. F. Bodon. A trie-based APRIORI implementation for mining frequent item sequences. *1st Int'l Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, pages 56–65, Chicago, Illinois, 2005.
3. C. Borgelt. Efficient Implementations of Apriori and Eclat. *Workshop of Frequent Item Set Mining Implementations*, Melbourne,FL, USA, 2003.
4. C. Borgelt. Recursion Pruning for the Apriori Algorithm. *2nd Workshop of Frequent Item Set Mining Implementations*, Brighton, UK, 2004.
5. J. Buhler, M. Tompa Finding Motifs Using Random Projections. *Journal of Computational Biology*, Vol. 9, No. 2, pages 225–242, 2002.
6. S. Dzeroski, I. Slavkov, V. Gjorgjioski, J. Struyf. Analysis of Time Series Data with Predictive Clustering Trees. *5th Int'l Workshop on Knowledge Discovery in Inductive Databases*, pages 47–58, Berlin, Germany, 2006.
7. P.G. Ferreira, P.J. Azevedo, C.G. Silva, R.M.M. Brito. Mining Approximate Motifs in Time Series. *9th Int'l Conf. on Discovery Science*, Barcelona, 2006.
8. P.G. Ferreira, P.J. Azevedo. Protein Sequence Classification through Relevant Sequence Mining and Bayes Classifiers. *12th Portuguese Conf. on AI*, 2005.
9. M.E. Futschik, B. Carlisle. Noise-Robust Soft Clustering of Gene Expression Time-Course Data. *Bioinformatics and Computational Biology*, 3:965–988, 2005.
10. W. Gaul, L. Schmidt-Thieme. Mining Generalized Association Rules for Sequential and Path Data. *IEEE ICMD* , pages 593–596, San Jose. 2001.
11. C. Gruber, M. Coduro, B. Sick. Signature Verification with Dynamic RBF Networks and Time Series Motifs. *10th Int'l W'shop on Frontiers in Handwriting Recognition*, 2006.

12. J. Hipp, A. Myka, R. Wirth, U. Gntzer. A new algorithm for faster mining of generalized association rules. *PKDD*, pages 74–82, Nantes, France, 1998.
13. K.L. Jensen, M.P. Styczynski, I. Rigoutsos, G.N. Stephanopoulos. A generic motif discovery algorithm for sequential data. *Bioinformatics.* 22:21–28, 2006.
14. E.J. Keogh, M.J. Pazzani. Scaling up Dynamic Time Warping for Datamining Applications. *KDD* , pages 285 – 289, Boston, Massachusetts, USA, 2000.
15. V. Kunik, Z. Solan, S. Edelman, E. Ruppin, D. Horn. Motif Extraction and Protein Classification. *IEEE Computational Systems Bioinformatics Conf.*, 2005.
16. T. Knorr. *Motif Discovery in Multivariate Time Series and Application to Hemodialysis Treatment Data.* MSc-Thesis, Albert-Ludwigs-Univ., Freiburg, 2006.
17. T. Knorr. Identifying Patients at Risk: Mining Dialysis Treatment Data *2nd German Japanese Symposium on Classification*, Berlin, 2006.
18. J. Lin, E. Keogh, S. Lonardi, B. Chiu. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. *8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery* 2003.
19. R. Manmatha, T.M. Rath. Indexing of Handwritten Historical Documents - Recent Progress. *Symposium on Document Image Understanding Technology*, Greenbelt, MD, pages 77-85, 2003.
20. S. Marcel, J.R. Millan. Person Authentication Using Brainwaves (EEG) and Maximum A Posteriori Model Adaptation. In *IEEE Trans. on Pattern Analysis and Machine Intelligence* 29:743–752, 2007.
21. P. Patel, E. Keogh, J. Lin, S. Lonardi. Mining Motifs in Massive Time Series Databases *IEEE ICDM*, 2002.
22. J. Pei, J. Han, J. Wang, H. Pinto, Q. Chen, U. Dayal, M. Hsu. Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. *IEEE Trans. on Knowledge and Data Engineering*, 16:1424–1440, 2004.
23. I. Pramudiono, M. Kitsuregawa. FP-tax: Tree structure based generalized association rule mining. *ACM/SIGMOD Int'l W'shop on Research Issues on Data Mining and Knowledge Discovery*, pages 60–63, Paris, 2004.
24. C.A. Ratanamahatana, E. Keogh. Making Time-series Classification More Accurate Using Learned Constraints. *SIAM Int'l Conf. on Data Mining*, 2004.
25. C.A. Ratanamahatana, E. Keogh. Everything you Know about Dynamic Time Warping is Wrong. *SIGKDD W'shop on Mining Temporal and Seq. Data*, 2004.
26. T.M. Rath, R. Manmatha. Word image matching using dynamic time wrapping. *CVPR*, II:521–527, 2003.
27. H. Sakoe, S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoustics, Speech, and Signal Proc.*, ASSP-26:43–49, 1978.
28. R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. *EDBT*, Avignon, France, 1996.
29. K. Sriphaew, T. Theeramunkong. A new method for finding generalized frequent itemsets in generalizes association rule mining. *ISCC*, pages 1040–1045, Taormina, Italy, 2002.
30. K. Sriphaew, T. Theeramunkong. Fast algorithms for mining generalized frequent patterns of generalized association rules. *IEICE Transactions on Information and Systems*, E87-D(3), 2004.
31. D. Yankov, E. Keogh, J. Medina, B. Chiu, V. Zordan. Detecting Time Series Motifs Under Uniform Scaling. *KDD*, pages 844–853, San Jose, USA, 2007.