MSTS: A System for Mining Sets of Time Series

Georg Lausen, Iztok Savnik, Aldar Dougarjapov

Freiburg University
Institute for Computer Science
Am Flughafen 17
D-79085 Freiburg im Breisgau, Germany
{lausen,savnik,dougarja}@informatik.uni-freiburg.de

Abstract

A system to support the mining task of sets of time series is presented. A model of a set of time series is constructed by a series of classifiers each defining certain consecutive time points based on the characteristics of particular time points in the series. Matching a previously unknown series with respect to a model is discussed. In contrast to a previously presented algorithm [17], the proposed algorithm finds the match with maximal precision in worst case time complexity $O(n \cdot m)$, where n is the length of a training series and m is the length of the test series. The architecture of the *MSTS*–System (*M*ining of *S*ets of *T*ime *S*eries) is described. As a distinctive feature the system is implemented as a database application: time series and the models, i.e. series of classifiers, are database objects. As a consequence of this integration, advanced functionality as the manipulation of models and various forms of meta learning can be easily build on top of MSTS.

Keywords: Time-series, data mining, knowledge discovery, and databases.

1 Introduction

Data in form of time series represents the behaviour of some time-dependent processes and can be used to exhibit the characteristics of the observed processes. In the literature there exists much work treating various aspects of time series. Matching characteristic patterns in time series databases is discussed in [2, 8, 1, 12, 5, 14]. These methods vary in the representation techniques for time series, the algorithms for measuring similarity between the time series, and in the search mechanisms used for mining patterns. Methods for the discovery of rules [6, 11] and frequent episodes [15] in time series have been proposed, as well. Rules and episodes describe the interrelations among the local shapes of time series.

The discovery of common patterns from a *set* of time series has a long tradition in statistics and one of the most widely used techniques is cross-correlation [3]. In the machine-learning framework the problem has previously been studied only by [13], to the best of our knowledge. Similar to cross-correlation, [13] base their approach on time series as a whole and classify them according to a distance measure. We take a different approach and classify concrete time points based on their local properties, i.e. properties of time points within a certain window on the whole time series. In contrast to the more mean-value oriented known technique, the resulting pointwise classification of a time series gives direct handles to explanations of the behaviour of the time series. This approach originated from a practical project to date tree trunks according to the yearly radial increase [17]; in the current paper we present a simplified and more efficient version of the algorithm and discuss its usuage as part of a data mining system.

Our current contributions are as follows. First we address the problem of finding the common characteristics of the time series which describe a set of similar processes. Given a description of a domain in the form of a set of training instances (time series), a model is constructed which describes the characteristic properties of the domain. The representation of the model is based on the "local" properties of time points — the time points are described by the characteristics of values which occurred close to the described time point. To build the model a sequence of classifiers is derived (computed by C5.0 [18]), which are represented by corresponding sets of rules. An algorithm for matching a sequence of values with the sets of training time series is introduced and analyzed. This task is also called *dating* the sequence.

One important direction during the last years is the development of data mining suites to support the mining tasks by integrating different mining techniques into a common framework and supporting interoperability with a database management system (e.g. [4]). Orthogonal to this stream of work we have implemented a system with the aim to provide support not for mining in general, but for the specific task of mining for applications interested in mining sets of time series. As one example of such applications we mention experimental results concerning the analysis of tree-ring time series which represent the yearly radial increases of tree trunks [17]. The architecture and implementation of *MSTS*–System (*M*ining of *S*ets of *T*ime *S*eries) is described. As a distinctive feature, MSTS is a database application: time series and the models, i.e. series of classifiers, are database objects. MSTS has been implemented under Windows NT using C++ as programming language and the Oracle 8 database environment.

Finally, we outline the benefits of our database-centered approach. Having the domain data and the domain models in a uniform database framework not only makes it easy for the miner to choose under several precomputed classifiers for the mining task, but also provides a powerful framework to manipulate models and to perform meta learning.

The structure of the paper is as follows. Section 2 presents the formal framework for mining and section 3 an algorithm for matching. Section 4 contains the architecture of MSTS and in particular presents an ER-schema specifying the content of the database. Moreover, real world experimental results underlining the usefulness of the approach are presented. In section 5 we conclude the paper by discussing advanced functionality of MSTS.

2 Formal Framework

The following is based on [17]. A time series $s = (v_1, \ldots, v_n)$ is a sequence of real numbers v_i representing the values of a time–dependent parameter of a process. The interval between subsequent time points is assumed to be constant. A set of time series is called a *domain* $\mathcal{D} = \{s_1, \ldots, s_m\}$. Each time series $s_i \in \mathcal{D}$, $s_i = (v_{i1}, \ldots, v_{in})$ has n values with time points $t(v_{ij}) = t_j$. Let $s_u = (u_1, \ldots, u_r)$ be a sequence of values where the time points $t(u_i)$ are not known. The time series of the domain are the training cases to build a model of the domain and s_u is a test case which has to be *dated*, i.e., whose pattern has to be matched against the model in order to predict the start time point $t(u_1)$.

The values of a time series $s \in \mathcal{D}$ in subsequent time points $t_j, \ldots, t_{j+k}$, $k \geq 0$, $j \geq 1$ is the projection of s on the given sequence of time points $s[t_j..t_{j+k}] = (s[t_j], \ldots, s[t_{j+k}])$. Let $w \geq 1$ and w odd, then the projection $s[t_j..t_{j+w-1}]$ is called a *window* of s and w is the size of the window.

The main idea later used for the construction of a model is to describe the characteristics of each particular time point in the domain by the properties of the values which are local with regards to the described time points. In other words, a time point t_j can be described by the characteristics of values which appeared some time before t_j, the values which appeared at t_j and some time after t_j. To this end, in the sequel, we consider sets of windows of the form $\mathcal{D}[t_{j-\lfloor w/2 \rfloor}..t_{j+\lfloor w/2 \rfloor}]$ $= \{s_i[t_{j-\lfloor w/2 \rfloor}..t_{j+\lfloor w/2 \rfloor}] | i \in [1..m] \wedge s_i \in \mathcal{D}\}$, where w is the size of the windows, $j > \lfloor w/2 \rfloor$ and t_j the *central point* of the window. Observe that any two windows of the same time series with central points t_l, t_{l+1} will overlap in $w-1$ time points.

Let $s_i[t_{j-\lfloor w/2 \rfloor}..t_{j+\lfloor w/2 \rfloor}]$ be a window with central point t_j. A window extended with the value of its central point is called a *dated* window. If $\mathcal{D}[t_{j-\lfloor w/2 \rfloor}..t_{j+\lfloor w/2 \rfloor}]$ is a given set of windows, then $\mathcal{D}^*[t_{j-\lfloor w/2 \rfloor}..t_{j+\lfloor w/2 \rfloor}]$ is the corresponding set of dated windows. A range of sets of dated windows is used as the input dataset for the construction of classifiers. Formally, a *dataset* S is defined as $S = \bigcup_{j \in [b..(b+d-1)]} \mathcal{D}^*[t_{j-\lfloor w/2 \rfloor}..$

$t_{j+\lfloor w/2 \rfloor}]$ where each window is dated, $w \geq 1$ is the window size, $d \geq 1$ is the size of the dataset given by the number of central points of the windows, and $b \geq \lfloor w/2 \rfloor + 1$ is the index of the central time point of the first window. Each window in a dataset is used for building a classifier as follows. The central point of the window denotes the class and all values constituting the window are the observed parameter values at the respective time points describing the corresponding class.

To simplify notation, throughout the paper we will assume $\alpha \cdot d = n - w + 1$ for some natural number α. The time series from $\mathcal{D}$ can now be split into a sequence of α datasets $S_1, \ldots, S_\alpha$, which are used for the definition of α classifiers $C_1, \ldots, C_\alpha$, where each classifier C_i, $1 \leq i \leq \alpha$ is to predict time points out of the interval $[\lfloor w/2 \rfloor + 1 + (i-1) \cdot d,\ \lfloor w/2 \rfloor + i \cdot d]$. Note that any two datasets S_i, S_{i+1} overlap in $\lfloor w/2 \rfloor$ time points, however any two corresponding classifiers C_i, C_{i+1} do not overlap with respect to their predicted time points, $1 \leq i < \alpha$. Computing a classifier given a dataset, in principle, is a standard mining task; however, as we have to compute $O(n)$ classifiers, the implementation of this task needs some futher discussion which is presented in Section 4.2.

3 Matching algorithm

Let $\mathcal{D} = (s_1, \ldots, s_m)$ be a domain such that the time series s_i are defined for the time points $t_1, \ldots, t_n$. Further, let $C_1, \ldots, C_\alpha$ be the set of classifiers constructed from α datasets of size d containing windows of size w. Finally, let $s_u = (u_1, \ldots, u_r)$, $r \geq w$, be a sequence of values which is to be dated. The task of matching then is to find the time point $t_u \in [t_1..t_n]$ of the first value u_1 of the sequence s_u such that the sequence s_u matches with $\mathcal{D}$ starting at t_u with a maximal precision. To this end all possible windows of length w are extracted from the input sequence s_u. Let $U = (U_1, \ldots, U_{r-w+1})$ be the sequence of possible windows of size w for s_u.

The matching algorithm we shall present simplifies and optimizes the approach proposed in [17]. While the algorithm in [17] tries to find a good guess for a prediction based on a threshold parameter stating a lower bound for the required probability of the guess, we directly classify each window with each classifier and store the results in a way, such that it can be guaranteed, that the time point with maximal collected probability will always be found. For length of the training sequences n and length of the test sequence m, we compute $O(n \cdot m)$ predictions. In contrast, the previous algorithm overall computes $O(\gamma \cdot n \cdot m^2)$ predictions, where $\frac{1}{m} \leq \gamma \leq 1$ depending on the threshold. Small thresholds imply large γ, i.e. a threshold of 0 implies $\gamma = 1$. This means a worst case time complexity of $O(n \cdot m^2)$ of the previous algorithm. Moreover, for a threshold greater 0 it is not guaranteed whether the time point with maximal precision will be found. This may happen because as precision the average of probabilities is computed, such that a high probability in one situation does not necessarily imply a high average.

The matching algorithm (cf. figure 1) classifies each of the windows in U with each available classifier (line 4). As the classifier predicts the central point of each

Algorithm Match

Input: A sequence $s_u = (u_1, \ldots, u_r)$ with corresponding
sequence of windows $(U_1, \ldots, U_{r-w+1})$ and
a sequence of classifiers $(C_1, \ldots, C_\alpha)$.

Output: Time point $t_i \in [t_1 \ldots t_n]$, which has been predicted
with the maximal collected probability.

Method:

1. **foreach** $(t_i \in [t_1 \ldots t_n])$ $ColProb[t_i] = 0$;
2. **foreach** (window $U_i \in (U_1, \ldots, U_{r-w+1})$)
3.   **foreach** (classifier $C_j \in (C_1, \ldots, C_\alpha)$) **do**
4. $(t, prob)$ = predictCentralPoint(U_i, C_j);
5. $firstPoint$ = $t - \lfloor w/2 \rfloor - i + 1$;
6. $ColProb[firstPoint]$ = $ColProb[firstPoint] + prob$;
7.   **od**;
8. **foreach** $(t_i \in [t_1 \ldots t_n])$ $ColProb[t_i] = ColProb[t_i]/(r - w + 1)$;
9. **return**($t_i \in [t_1 \ldots t_n]$ with maximal collected probability);
10. **end**;

Figure 1: Matching algorithm.

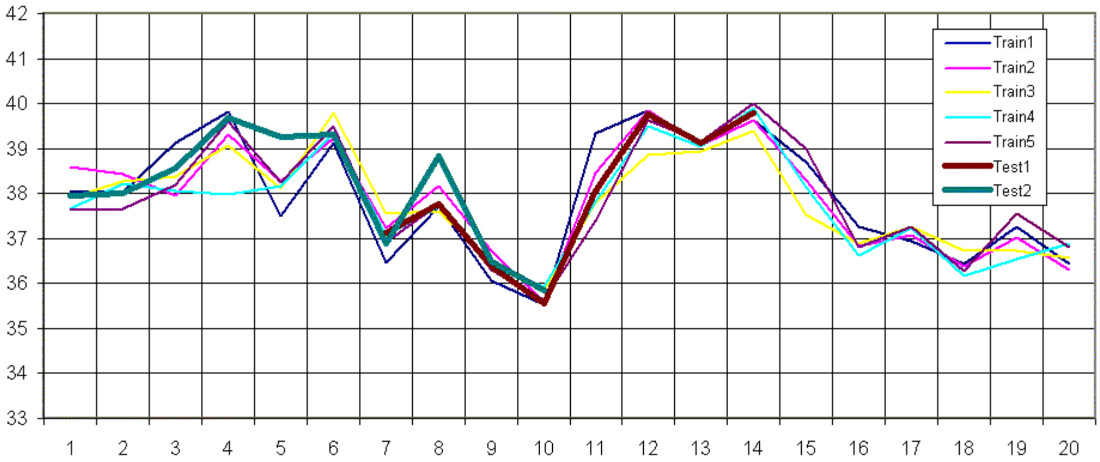


Figure 2: Graph of a set of 5 time series and two test sequences.

window, the predicted central point is transformed into the corresponding predicted initial time point of s_u (line 5). `ColProb` then sums up the probabilities a certain time point is predicted to be the initial time point of s_u (line 6). Finally, the algorithm computes for each time point t_i the respective *collected* probability, which is computed as the average of the probabilities with respect to the number of classifications, which could predict t_i. This number is given by the number of windows in U, i.e. $(r - w + 1)$. To see this assume the contrary, i.e. one time point is predicted more often than the number of windows in s_u. However this is not possible, because different classifiers predict disjoint intervalls. The final output of the algorithm then is the time point t_i, for which the collected probability is maximal; collected probabilities are our measure for precision.

To demonstrate the steps of the algorithm we will investigate two test sequence. Test sequences s_u^1 and s_u^2 and a given set of 5 training sequences are shown in figure 2. While the shape of s_u^1 was chosen such that a start point of 7 could be expected, s_u^2 was constructed by first trying to capture the shape of the series between 1 and 10, but then changing the contour of the series at time point 5 and 8 significantly.

Assume a window size $w = 5$ and a size of the dataset $d = 4$. Applying C5.0 to compute the classifiers and to predict the start points of the test sequences allows us to trace the computation for the matching tasks. For each window w and each classifier C (by writing $Ci..j$ we express that the classifier can predict as *central* point of a window a time point between i and j) the predicted first point of the considered test case s_u and the respective probability is given. Because of the underlying very small set of training examples the probabilities shown in the tables cannot be interpreted in a strong statistical sense. Results of a real practical experiment are reported in section 4.3.

Matching of s_u^1:

window	C3..6	prob	C7..10	prob	C11..14	prob	C15..18	prob
w_1	4	0.86	**7**	0.86	11	0.86	13	0.86
w_2	1	0.86	**7**	0.86	10	0.86	15	0.86
w_3	0	0.86	6	0.86	**7**	0.86	13	0.86
w_4	-1	0.86	2	0.86	**7**	0.86	10	0.86

Matching of s_u^2:

window	C3..6	prob	C7..10	prob	C11..14	prob	C15..18	prob
w_1	2	0.86	5	0.86	**11**	0.86	13	0.86
w_2	1	0.86	4	0.86	**11**	0.86	12	0.86
w_3	1	0.75	3	0.43	9	0.86	**11**	0.86
w_4	-1	0.86	2	0.43	9	0.86	**11**	0.86
w_5	-1	0.75	1	0.43	7	0.86	9	0.86
w_6	-1	0.86	1	0.86	6	0.86	9	0.43

We summarize by mentioning the best, the second and third best matching starting point:

	s_u^1	ColProb	s_u^2	ColProb
Bestmatch	**7**	0.86	**11**	0.57
next best	10	0.43	9	0.5
next best	13	0.43	1	0.48

As expected, the start point of s_u^1 is predicted by 7; for s_u^2 the algorithm predicts start point 11 which, in fact, begins a shape of the training series which is similar to the test series s_u^2. However, here the situation is not as supported by the collected probabilities as in the previous case. The collected probabilities show, that start points 1, 9, 11 are more or less equally supported. However, again we have to emphasize, that because of the very small set of training cases the computed probabilities must be taken with great care and should not be considered to be statistically sound!

4 MSTS: Mining Sets of Time Series

4.1 Architecture

One important direction during the last years is the development of data mining suites to support the mining tasks by integrating different mining techniques into a common framework and supporting interoperability with a database management system (e.g. [4]). We will now present the architecture of the *MSTS*–System (*M*ining of *S*ets of *T*ime *S*eries); a detailed description can be found in [7]. In contrast to general data mining suites MSTS aims at supporting the specific task of mining sets of time series and therefore can be considered as complementing a mining suite. The mining algorithms of MSTS are based on the techniques described in section 2, 3. For building classifiers C5.0 [18] is used. The training cases for the classifiers are given by the datasets derived from a given domain. The attribute values of each case in a window of the dataset are the values of the observed parameter at a respective time points. The class to be trained is the value of the central point of the windows.

Working with MSTS supports modelling of sets of time series (called *sequences*) and matching undated sequences against a model. The sequences, resulting models and matchings are database objects. Further database objects are sessions. Sessions are the logical unit a user is working with MSTS. For a domain given by a set of sequences, different models may exist in the database. This allows to explore several alternatives when matching unknown series. Different models may arise because of varying window sizes and sizes of the datasets. Grouping modelling and matching tasks into sessions allows the user to keep track of several explorations of the same, or overlapping domains.

The information content of the database underlying MSTS is specified by the ER–schema shown in figure 3. Objects[1] of type `sequence, model` and `session` can be created, deleted or viewed; objects of type `match` can only be viewed because they are computed by the system. Objects of type `sequence` are further classified in `dated` and `undated`, where in the first case we further distinguish `sure` and `estimated`. Estimated sequences are those whose dating has been performed by MSTS. An object of type `domain` then is given by a set of dated sequences. A model is defined by a set of classifiers and related to exactly one domain. This is represented by object types `model` and `classifier` and relationship types `shape` and `part_of`. A classifier may be part of more than one model, however these must be shapes of the same domain; therefore a classifier is related by `owned by` with exactly one domain. Matches are represented by relationship type `match`; each match is further characterized by a set of submatches, which means the elements of the table relating windows of the test sequence and the classifiers of the used model. This gives rise to object type `submatch` and relationship type `contain`. The remaining part of the ER–schema is self-explanatory. Because of limited space attributes of the types are not discussed; we only mention that `classifier` has two attributes to store the rules computed by C5.0 in binary format, such that they can be reused for prediction,

[1] We talk about *objects* instead of *entities*.

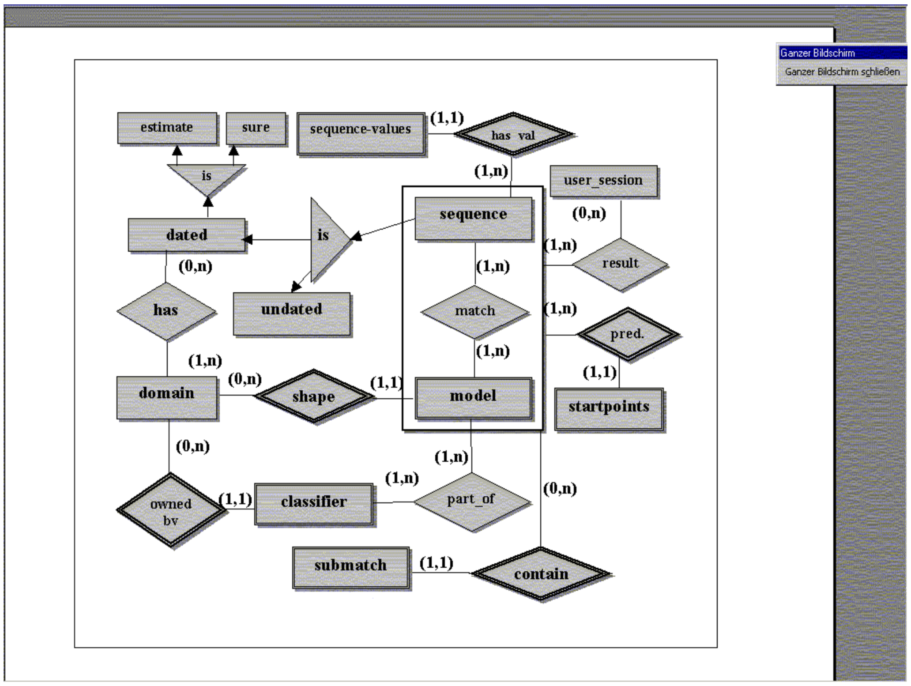


Figure 3: ER–Schema of the database underlying MSTS; attributes are omitted.

and ascii format, such that they can be shown to the user.

4.2 Implementation

MSTS is implemented as a database application written in C++ and connected to a Oracle 8 database server by ODBC. The coupling with the database is based on cache memory which is carefully maintained by MSTS. With respect to the sequences there is a ternary relation `SEQUENCE_VALUES` stored in the database, which relates a certain sequence by its ID to a certain time point giving the respective value of the parameter of interest.

When a range of classifiers has to be computed the corresponding datasets are extracted from the database as follows. The database receives an SQL-expression which extracts from `SEQUENCE_VALUES` the relevant subset of rows which are needed to compute the cases (windows) for constructing one dataset. This subset is sorted by time and with respect to a certain time point by sequence ID. Using only one database cursor the input file for C5.0 now can be easily derived in C++. As consequtive datasets overlap, when switching from one dataset to the next only the missing rows are extracted from the database. Note that because of sorting first by time and within one time point by sequence ID, the overlapping already sorted part can be reused. As only one database cursor is used and each row from the database is involved in exactly one sort operation, the work to build the input of C5.0 is minimized.

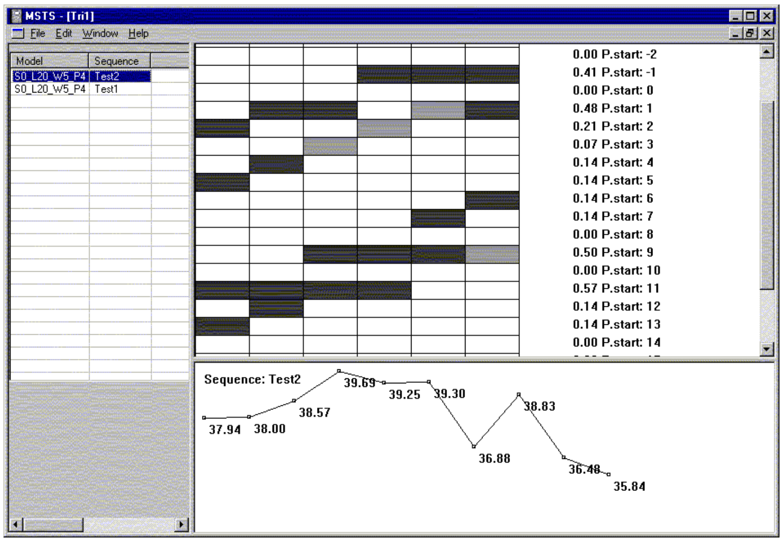
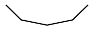


Figure 4: MSTS screendump demonstrating the mining task.

4.3 Experimental Results

MSTS has been successfully applied for the analysis of tree-ring time series which represent the yearly radial increases of tree trunks. The results presented in [17] are also valid for MSTS and will be mentioned shortly.[2] The domain was build out of 50 tree-ring time series of length 100 to 250. A typical example of a set of rules describing a time point 1900 is as follows (window size 21):

```
plus7 > 3.086, plus9 > 2.693, plus10 > 3.071, plus14 > 3.526 -> class 1900 [0.965]
minus10 > 2.861, plus10 <= 3.071, plus13 <= 3.267 -> class 1900 [0.916]
minus5 <= 2.835, minus3 <= 2.979, plus10 > 3.071 -> class 1900 [0.748]
```

Here `plus`i, respectively `minus`i, refers to the time point 1900+i, respectively, 1900-i. The probabilities of the rules are given in brackets. A first analysis based on inspecting the rules shows, that time point 1910 seems of some importance for predicting time point 1900. This observation could be the starting point of a deeper investigation. As an example for matching consider the following sequence: . For a model of the domain with a window of size 5, the following is the result of matching: Number of matches = 8; Average probability = 0.89; Starting time points: 1880,1888,1896,1912,1918,1929,1937,1963.

5 Conclusion

In this paper we have presented MSTS, a system to mine sets of time series. As a distinctive feature, MSTS stores not only the time series, but also the classifiers and the information used to construct them in the database. This gives us the opportunity to reason about the mining process using as a high-level language SQL.

[2]The detailed analysis reported in [17] is based on a prototypical implementation in Perl.

However we can also go beyond SQL.

It has been recently pointed out that the integration of inductive and deductive reasoning is a promising direction [9, 10]. The underlying motivation is to control the overall mining task and to incorporate background knowledge. We also support this direction, however are convinced, that a less ambitious approach based on commercial platforms (C++, Oracle 8) is also of interest.

MSTS can be extended towards the above goals, because sequences, classifiers and models are database objects and thus can be easily manipulated. Because of limited space we only sketch three possible extensions. Consider a model given by a range of classifiers over time points $[t_i..t_j]$. Because each classifiers defines time points by rules, it is easy to compute the *projection* of the model to $[t'_i..t'_j]$, where $t_i \leq t'_i \leq t'_j \leq t_j$. Similarly, if there are models on time points $[t_{i1}..t_{j1}]$, $[t_{i2}..t_{j2}]$, where $t_{j1} = t_{i2}$ and the size of the windows are equal, then the *union* can be computed. The next extension is a form of *meta learning* called *combining* [16]. Here the predictions of the *base* classifiers performed on their validation sets form the basis for the meta-learner's training set. In our scenario the role of a classifier is taken by a model. One interesting aspect here then is the possibility to predict based on different window sizes at the same time. This allows to combine models based on small windows appropriate to reflect short term effects in the time series with models based on large window sizes appropriate to reflect long term effects. It is easy to see that the ER-schema of MSTS shown in Figure 3 provides the necessary information: a match is a relationship between a sequence and a model, which is further described by the determined startpoints. For one sequence there may exist several matches which then may form the meta-learner's training set.

We are planning to extend MSTS according to these lines and to systematically compare our approach with other traditional techniques, in particular those based on cross-correlation, nearest-neighbors, and Euclidean distance.

6 Acknowledgement

We would like to thank Heinrich Spieker, Hans-Peter Kahle and Sebastian Hein for many insightful discussions, in particular for giving us the opportunity to use real world data. We are also gratefully indebted to Luc de Raedt and Stefan Kramer for valuable comments on our work.

References

[1] Agrawal, R., Lin, K.I., Sawhney, H.S., Shim, K., 'Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-Series Databases', In *Proc. of VLDB Conference*, pp.490-501, 1995.

[2] Berndt, D.J., Clifford, J., 'Finding Patterns in Time Series: A Dynamic Programming Approach', In *Advances in Knowledge Discovery and Data Mining*, U.M. Fayyad, Shapiro, G.P., Smith, P., and Uthurusamy, R., Eds., MIT 1996.

[3] Box, G.E.P., Jenkins, G.M., 'Time Series Analysis - Forecasting and Control', San Francisco: Holden-Day, 1970.

[4] http://www.spss.com/clementine/

[5] Das, G., Gunopulos, D., Mannila, H., 'Finding similar time series', In *Proc. of PKDD'97*, LNCS, 1997.

[6] Das, G., Gunopulos, D., Mannila, H., 'Rule discovery from time series', In *Proc. of Conf. on Knowledge Discovery and Data Mining*, AAAI Press, 1998.

[7] Dougarjapov, A., 'MSTS: Mining Sets of Time Series', *Master Thesis (in preparation)*, Institut für Informatik, Universität Freiburg, 2000.

[8] Faloutsos, C., Ranganathan, M., Manolopoulos, Y., 'Fast Subsequence Matching in Time-Series Databases', In *Proc. of SIGMOD Conference*, 1994.

[9] Giannotti, F., Manco, G., Nanni, M., Pedreschi, D., Turini, F., 'Integration of Deduction and Induction for Mining Supermarket Sales Data', In *Proc. PADD'99, Practical Application of Data Discovery, London*, 1999.

[10] Giannotti, F., Pedreschi, D., 'Knowledge Discovery and Data Mining', In Tutorial Slides, EDBT 2000, 2000.

[11] Guralnik, V., Wijesekera, D., Srivastava, J., 'Pattern Directed Mining of Sequence Data', In *Proc. of Conf. on Knowledge Discovery and Data Mining*, AAAI Press, 1998.

[12] Keogh, E., Smyth, P., 'A Probabilistic Approach to Fast Pattern Matching in Time Series Databases', In *Proc. of Conf. on Knowledge Discovery and Data Mining*, AAAI Press, 1997.

[13] Keogh, E.J., Pazzani, M.J, 'An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback', In *Proc. of Fourth Conf. on Knowledge Discovery and Data Mining*, 1998.

[14] Keogh, E.J., Pazzani, M.J, 'An Indexing Scheme for Fast Similarity Search in Large Time Series Databases', In *Proc. of Conf. on Scientific and Statistical Database Management*, 1999.

[15] Mannila, H., Toivonen, H., Verkamo, A.I., 'Discovery of frequent episodes in event sequences', *Data Mining and Knowledge Discovery Journal*, Vol.1, No.3, pp. 259-289, Nov 1997.

[16] Prodromidis, A.L., Chan, P.K., Stolfo, S.J., 'Meta-Learning in Distributed Data Mining Systems: Issues and Approaches', To appear in *Advances in Distributed Data Mining*, Kargupta and Chan (eds.), AAAI Press, 1999.

[17] Savnik, I., Lausen, G., Kahle, H.-P., Spiecker, H., Hein, S., 'Algorithm for Matching Sets of Time Series', Technical Report, DBIS, Institut für Informatik, Universität Freiburg, Mar 2000, http://www.informatik.uni-freiburg.de/ dbis/Publications/2K/match.ps

[18] Quinlan, J.R., *C4.5: Programs for Machine Learning*, Morgan Kauffman, 1993.