SVM Kernels for Time Series Analysis*

Stefan Rüping**

**CS Department, AI Unit, University of Dortmund, 44221 Dortmund, Germany, E-Mail stefan.rueping@uni-dortmund.de

Abstract. Time series analysis is an important and complex problem in machine learning and statistics. Real-world applications can consist of very large and high dimensional time series data. Support Vector Machines (SVMs) are a popular tool for the analysis of such data sets. This paper presents some SVM kernel functions and disusses their relative merits, depending on the type of data that is used.

Keywords. Support Vector Machines, Time Series

1 Introduction

Time is a phenomenon which is both very complex and very important in many real-world problems. Its importance comes from the fact that almost every kind of data contains time-dependent information, either explicitly in the form of time stamps or implicitly in the way that the data is collected from a process that varies with time (e. g. a machine that is getting worn out, sales that are influenced by changing tastes or changing contents of web sites). A reason for its complexity is that time can be represented in a multitude of different representations. As always, an effect that is obvious in one representation may be very much hidden in another representation. In [8] Morik gives a discussion of different representations and learning tasks for time phenomena.

Of all the possible representation of time, time series, i. e. the representation of a time dependent observation x_t at (usually equidistant) time points t as tuples (t, x_t), are the most common. This is the case because it is the easiest way to aquire time dependent data - all you need to have to gather the data is a clock. Also, there are many statistical algorithms that can be used on numerical time series data, e. g. ARIMA modeling or Fourier transforms.

One of the main problems of time series analysis, the forecasting of time series, can be very easily stated as a pure numerical problem: Split the time series $x_1, \ldots, x_N$ into windows $(x_i, \ldots, x_{i+k-1})$ of size k. Then find a function $f : \mathbf{R^k} \rightarrow \mathbf{R}$ such that $f(x_i, \ldots, x_{i+k-1}) = x_{i+k}$ for every $i \in \{0, N-k\}$. Other learning tasks, such as classification or similarity com-

* This paper has already been published in the Proceedings of the LLWA 01, University of Dortmund, 2001

putation of time series can also be formulated as purely numerical problems. Support Vector Machines (SVMs, see section 2) have been successfully applied for this kind of learning tasks ([10], [9], [5]).

So if we have a purely numerical problem and purely numerical data, what is the problem with a purely numerical algorithm? The reason is that the real-world process, which lies behind the data, in general will not be this simple.

See the following example: Suppose we are given the weekly sales of candles in some store and we want to use machine learning to predict how many candles will be sold next week. Obviously, there will be an extremely high peak of sales at Christmas time. The usual (numerical) solution would be to notice that there is a cycle of one year in the data, so one could try to use the sales of one year ago to predict the next sales - but as the data is given per week, there could be 51 or 52 weeks between two Christmas's, depending on the actual date. The problem gets much worse for easter instead of Christmas, because the date of the easter holidays can vary about six weeks.

As another example, think of the problem of analyzing data in intensive care medicine. A time series in this field, for example some blood pressure, is characterized by high variation, which is due to normal physiological effects and small variations in the sensors. Such a time series may exhibit three different behaviors: It could be stable, which in this case means that the variation is not too high, it could have one or several outliers, i. e. observation that lie out of the normal variation but which are not significant of the state of the patient (e. g. measurement errors) or it may be a significant change in the time series, i. e. the structure of the time series itself changes. To decide which of the three cases a given observation belongs to, is very complicated and may depend on many facts, including very high level medical reasoning.

As we see, there is always a gap between numerical analysis and high-level, symbolic reasoning that needs to be bridged. To incorporate higher level reasoning and background knowledge into the analysis of numerical time series there are two possible ways:

1. Bring the data to the high-level reasoning: Transform the time series into a representation more suitable for higher level reasoning, e.g. discretize the time series and apply some logical modeling.

2. Bring the high-level reasoning to the data: Choose the hypothesis space and transform the data for the numerical learner in such a way, that results that are meaningful in some way can be found and are preferred. For examples, do some higher level analysis of the data and use the results as additional features for the numerical algorithm, like flags for the occurrence of special holidays in the sales data, or choose a hypothesis space that corresponds to a meaningful model.

This paper deals with the second approach. In the context of Support Vector Machines, kernel functions (which define the hypothesis space) are discussed that can be interpreted as some kind of time series model. Experiments are made to discover if these different model assumptions have effects in practice and if there exist kernel functions that allow time series data to be processed with Support Vector Machines without intensive preprocessing.

The remainder of this paper is organized as follows: In the next section, a short introduction to Support Vector Machines will be given. Section 3 presents some kernel function and discusses the ideas about time series that lies behind them. In Section 4, experiments are made to see how these kernels perform on real-world data.

2 Support Vector Machines

Support Vector Machines (SVMs) are based on the work of Vladimir Vapnik in statistical learning theory [15]. Statistical learning theory deals with the question, how a function f from a class of functions $(f_\alpha)_{\alpha\in\Lambda}$ can be found, that minimizes the expected risk

$$R[f] = \int\int L(y, f(x))dP(y|x)dP(x) \tag{1}$$

with respect to a loss function L, when the distributions of the examples $P(x)$ and their classifications $P(y|x)$ are unknown and have to be estimated from finitely many examples $(x_i, y_i)_{i\in I}$.

The SVM algorithm solves this problem by minimizing the regularized risk $R_{\text{reg}}[f]$, which is the weighted sum of the empirical risk $R_{\text{emp}}[f]$ with respect to the data $(x_i, y_i)_{i=1\ldots n}$ and a complexity term $||w||^2$

$$R_{\text{reg}}[f] = R_{\text{emp}}[f] + \lambda||w||^2.$$

In their basic formulation, SVMs find a linear decision function $y = f(x) = \text{sign}(w\cdot x + b)$ that both minimizes the prediction error on the training set and promises the best generalization performance. Given the examples $(x_1, y_1), \ldots, (x_n, y_n)$ this is done by solving the following optimization problem:

$$\begin{aligned} \Psi(w,\xi,\xi^*) &= \frac{1}{2}(w^T w) + C\sum_{i=0}^{n}\xi_i \\ &\to \min \end{aligned} \tag{2}$$

subject to

$$y_i(w^T x_i + b) \leq 1 - \xi_i, i = 1, \ldots, n \tag{3}$$

$$\xi_i \geq 0, i = 1, \ldots, n \tag{4}$$

The hyperplane vector w has a representation in terms of the training examples $(x_i, y_i)_{i\in I}$ and their Lagrangian multipliers $(\alpha_i)_{i\in I}$, that are calculated during the optimization process:

$$w = \sum_{i\in I}\alpha_i y_i x_i.$$

3 Kernel Functions

One of the major tricks of SVM learning is the use of kernel functions to extend the class of decision functions to the non-linear case. This is done by mapping the data from the input space X into a high-dimensional feature space $\mathcal{X}$ by a function

$$\Phi : X \to \mathcal{X}$$

and solving the linear learning problem in $\mathcal{X}$. The actual function Φ does not need to be known, it suffices to have a kernel function k which calculates the inner product in the feature space.

$$k(x, y) = \Phi(x) \cdot \Phi(y)$$

It was noticed by Schölkopf in [14] that the kernel function defines a distance measure d on the input space by

$$\begin{aligned} d^2(x, y) &= (\Phi(x) - \Phi(y))^2 && (5) \\ &= k(x, x) - 2k(x, y) + k(y, y). && (6) \end{aligned}$$

This shows the kernel function $k(x, y)$ can be interpreted as a measure of similarity between the examples x and y.

3.1 Linear kernel

The linear kernel $k(x, y) = x \cdot y$ is the most simple kernel function. The decision function takes the form $f(x) = w \cdot x + b$. When one uses the linear kernel to predict time series, i. e. $x_T = f(x_{T-1}, \ldots, x_{T-k}) = \sum_{t=1}^{k} w_t x_{T-t} + b$, this means the resulting model is an statistical autoregressive model of the order k (AR[k]).

With this kernel, time series are taken to be similar, if they are generated by the same AR-model.

3.2 RBF kernels

Radial basis kernels take the form $k_\gamma(x, y) = \exp(-\gamma ||x - y||^2)$. clearly, the similarity of two examples is simply judged by their euclidian distance.

In terms of time series, this has a parallel in the so-called phase space representation. Assume the time series is generated by a function g such that $x_T = g(x_{T-1}, \ldots, x_{T-k})$. If one takes the time series $x_1, \ldots, x_k, \ldots, x_N$ and plots the $(k+1)$-dimensional vectors $(x_t, x_{t+1}, \ldots, x_{t+k})$, the resulting plot is a part of the graph of g, so the function g can be estimated from the time series (see Figures 1 and 2).

Especially, assuming that the function g is linear and the data is generated by $x_T = g(x_{T-1}, \ldots, x_{T-k}) + \eta$ where η is a Gaussian noise (i. e. the time series model is AR[1]), it can be shown that most of the data lies in an ellipsoid defined by the mean of the time series and the variance of η. In [1] this is used in the phase space procedure for finding outliers in the time series.

This shows that information about a window of a time series can be gotten from other windows of the time series that are similar in means of the euclidian distance, which makes the RBF kernel promising for time series analysis.

3.3 Fourier Kernel

A common transformation for the analysis of time series data is to use the Fourier transform (see Figure 3). This representation is useful if the information of the time series does not lie in the individual values at each time point but in the frequency of some events. It was noted by

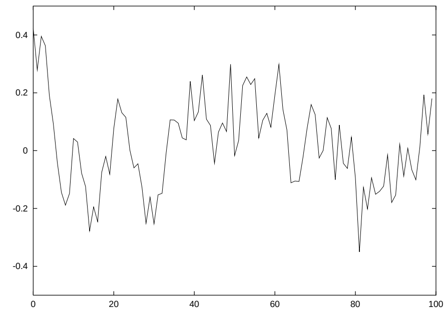

Figure 1   AR[1] time series.

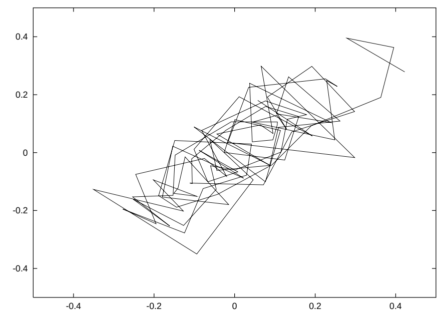

Figure 2   Phase space embedding of the time series in Figure 1.

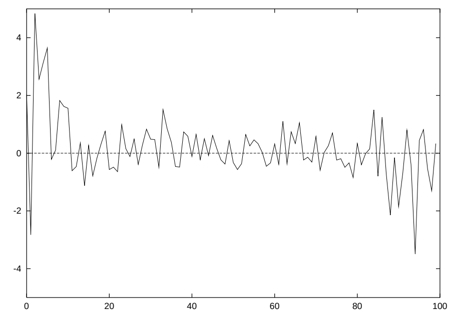

Figure 3   Fourier transform of the time series in Figure 1.

Vapnik [15] that the inner product of the Fourier expansion of two time series can be directly calculated by the regularized kernel function

$$k_F(x, y) = \frac{1 - q^2}{2(1 - 2q\cos(x - y)) + q^2}.$$

3.4 Subsequence Kernels

As mentioned in Section 1, time dependent processes may not show themselves by certain events happening at a fixed time-point, but by a certain sequence of events, independent of the actual time. In between this events, outliers or random observations may occur. Therefore, many algorithms for finding similar time series do not consider the whole time series but look for informative subsequences ([2], [4]).

A subsequence kernel for discrete sequences was used for text classification in [7]. However, the calculation of this kernel depends on the discreteness of the sequences, so it is not applicable to real-valued time series.

In section 3.2, radial basis kernels were used for time series analysis on the basis that similar time series (in means of the euclidian distance) should have the same properties. Based on the observation that time series can be viewed as similar if they have similar subsequences, a matching kernel function can be defined as

$$k_{subseq}(x, y) = \sum_{s_x, s_y} K_\gamma(s_x, s_y),$$

where s_x and s_y are subsequences of x and y of a fixed size. As each subsequence-part of the kernel will be close to zero for all non-matching subsequences, the kernel effectively is defined on only the matching subsequences of the time series.

As there are $\binom{n}{k}^2$ pairs (s_x, s_y) of length k in a time series of length n, for practical purposes one has to restrict the set of subsequences that are used. A possible solution is to use only connected subsequences, to use only subsequences with the same index set for x and y or to restrict k in some way. In the experiments in this paper, only $k = n - 1$ was used.

3.5 PHMM Kernels

One can take the idea of subsequences as a fitting representation of time series a bit further. The idea behind the subsequence representation is that there is a process hidden behind the data, which can only be observed at certain time points and is inactive or hidden behind noise the other times.

Hidden Markov Models [11] offer a model in which these assumptions are explicitly modeled. In a Hidden Markov Model, the output is assumed to be generated by a process, which is in one of finitely many states at each time. At every step, the process jumps from one state to the next state with a given transition probability, which is only dependent on the states. The state sequence itself cannot be observed, all that is known are the outputs that are generated by the process by a certain probability dependent on the state. For a given Hidden Markov Models and a sequence of observations, the probability that this sequence is generated by the model can be calculated.

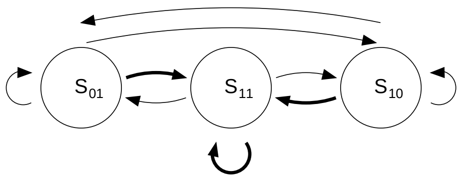

Figure 4   The PHMM used. Thick lines depict high transition probabilities.

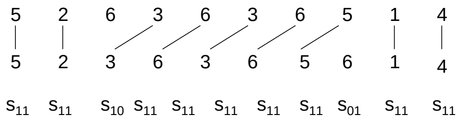

Figure 5   An example output sequence for a PHMM and corresponding state sequence.

Pair Hidden Markov Models (PHMMs) are Markov Models, which generate two output sequences simultaneously. At each state, either an output for the first sequence, an output for the second sequence or a common output for both sequences is generated. Probability estimation for PHMMs can be efficiently done in time $O(s^2k^2)$, where s is the number of states of the PHMM and k is the length of the example sequences (see [3]).

PHMMs for discrete valued series have been applied as SVM kernels in [16], where it has been shown that the joint probabilities of pairs of sequences defined by PHMMs indeed defines a proper kernel function. The extension to real valued sequences is very easily done by replacing the discrete probabilities by continuous probability densities, a standard trick for Hidden Markov Models. Similar work can be found in [6].

In the experiments in this paper, the very simple PHMM shown in Figure 4 has been used. The PHMM consists only of three states. In state s_{11} an output (x, y) for both sequences is generated such that $x - y$ is normally distributed. In state s_{10} (s_{01}) only a output for the first (second) sequence is generated. In all states, the transition probability to state s_{11} is λ (usually near 1) and the probability for a transition to any other state is $(1 - \lambda)/2$. Informally speaking, the idea behind this model is that two similar sequences usually should look the same, just that once in a while an extra output is generated in one of the sequences (see Figure 5).

It is also possible to learn the parameters of the PHMM (i. e. the transition and the output probabilities) from the data by the use of a modified Baum-Welch algorithm ([11], [3]). The problem is how to select the examples for learning: As the PHMM should assign high probabilities to similar time series and low probabilities to dissimilar ones, it must be trained on similar time series, so they have to be known beforehand. A possible solution could be to estimate the PHMM parameters on positive examples alone in the case of classification or to manually select similar time series. But of course this can easily lead to a skewed sampling of examples and hence to a bad parameter estimation. Because of this problems, no parameter estimation from data was done in the experiments of this paper.

3.6 Extension to Multivariate Time Series

Until now, the focus has only been on univariate time series. But how can multivariate series $(x_1^{(1)}, \ldots, x_1^{(d)}), \ldots, (x_T^{(1)}, \ldots, x_T^{(d)})$ be dealt with?

There is a simple trick that works on every case: The class of kernel functions is closed under addition and multiplication, so we can always build a multivariate model by addition or multiplication of univariate time series models. This model assumes that there is no interaction between the single time series $x_1^{(i)}, \ldots, x_T^{(i)}$.

For some kernels, there are also more complex ways to deal with this modeling problem. The RBF kernel $K(x, y) = \exp(-\gamma||x - y||^2)$, for example, can be used directly with x and y representing the multivariate time series windows. This model assumes a full interaction between the single time series and over all time points in the window, which may as easily be wrong as the assumption of no interaction. In particular, a single outlier in one time series on one time point will influence the whole example.

A similar approach can be used in the subsequence kernel. As well as looking for matching subsequences in the single time series, one can also search matching subsequences of the whole time series. As well as for the RBF kernel, the choice is between no intercation and complete interaction of all single time series.

In the PHMM kernel, the extension to the multivariate case can be done by defining a d-dimensional output probability on the states. Also, one can split up each of the states S_{11}, S_{10} and S_{01} into a number of states, where each state still produces only output for both, the first or the second sequence, respectively, but with different probabilities. This can be used to define several probabilities on the multivariate outputs. Therefore, the PHMM kernel can define a very complex model on the time series - on the cost of having to estimate the model parameters.

4 Experiments

To test the performance of these kernels on real-world data sets, some experiments were made. In all experiments, 10-fold cross-validation was used to get an estimation of the mean absolute error (MAE) and mean squared error (MSE) resp. the accuracy on these data sets. The SVM implementation mySVM [13] was used in the experiments.

4.1 Chromatography

This data comes from the chemical process of Chromatography. Chromatography is used in chemical industry to separate temperature sensitive substances. A mixture of components is injected into a column filled with porous particles. The component with the highest adsorption ability has the longest residence time in the column and the component with the lowest adsorption ability reaches the column end at first. The concentration of the components is measured over time and gives a time series which is characteristic of the components.

The learning task in this example is to identify the components by the approximation of a certain real-valued parameter called *Henry*, that is characteristic of the components.

At all there were 500 time-points for each curve. From these time-points, only 275 points had non-zero values and were used. To reduce the size of the data set further, only each k-th time-point for $k \in \{1, 5, 10, 30, 50\}$ was used, giving five different datasets with attributes set sizes of 275, 55, 27, 9 and 5 attributes.

4.1.1 Dot Kernel

Dataset	MAE	MSE
1	0.534	0.630
5	0.501	0.515
10	0.529	0.530
30	0.744	0.913
50	1.189	2.478

4.1.2 RBF Kernel

Previous investigation showed that usable values of the parameter γ of the RBF kernel were in the range of 0.001 to 0.1.

Dataset	gamma	MAE	MSE
1	0.001	0.201	0.452
1	0.01	0.417	0.896
1	0.1	1.124	3.029
5	0.001	0.189	0.215
5	0.01	0.189	0.344
5	0.1	0.493	1.079
10	0.001	0.239	0.166
10	0.01	0.175	0.237
10	0.1	0.321	0.622
30	0.001	0.609	0.713
30	0.01	0.309	0.305
30	0.1	0.257	0.390
50	0.001	1.015	1.865
50	0.01	0.591	0.682
50	0.1	0.386	0.571

4.1.3 Fourier Kernel

Dataset	q	MAE	MSE
1	0.25	2.058	6.161
1	0.5	2.059	6.140
1	0.75	2.059	6.140
5	0.25	2.046	6.117
5	0.5	0.815	1.065
5	0.75	2.059	6.140
10	0.25	0.559	0.464
10	0.5	0.461	0.222
10	0.75	2.059	6.140
30	0.25	0.300	0.123
30	0.5	0.399	0.183
30	0.75	0.466	0.214
50	0.25	0.435	0.407
50	0.5	0.406	0.323
50	0.75	0.400	.184

4.1.4 Subsequence Kernel

Dataset	MAE	MSE
1	-	-
5	0.578	1.574
10	0.584	0.776
30	1.082	2.322
50	1.805	4.397

This experiment shows the limitations of the subsequence kernel: The runtime on dataset 5, i. e. with 55 attributes, was in the range of several days. Therefore, the experiments with the even larger dataset 1 (275 attributes) were omitted. Clearly this type of kernel function can only be used for very low dimensional data.

4.1.5 PHMM Kernel

Dataset	MAE	MSE
1	-	-
5	0.786	1.853
10	0.589	1.285
30	0.359	0.738
50	0.488	0.998

The runtime of the SVM with the PHMM kernel was better than with the subsequence kernel, but still in the range of some days for the 55-attribute dataset. As in the case for the subsequence kernel, the experiments with the 275-attribute dataset were omitted.

All in all, the RBF kernel with $\gamma = 0.01$ on the 27-attribute dataset shows the best performance (MAE = 0.175). The Fourier kernel performs worse (MAE = 0.300), but still with good results (9 attributes, $q = 0.25$). The PHMM kernel on the 9-attribute dataset comes third (MAE = 0.359). The dot and subsequence kernels show the worst performance (MAE = 0.501 and MAE = 0.578, respectively).

In another experiment with this dataset [12] reports that even better results for this dataset were found by using a specially construction aggregate features of the time series such as location of the maximum and turning points. This shows that the important characteristic of a time series in this case is its similarity to others in its overall shape. This explains why the RBF kernel is suited for this task.

Also the bad performance of the subsequence kernel can be explained: The equidistant feature selection in these experiments together with the high reduction of the input dimensionality (from 275 to 9 in the extreme case) leads to a high gap between two successive time points in the time series. Therefore, the comparison of different time points and thus the use of the subsequence kernel is not reasonable.

4.2 Retail Store Data

This data consists of the weekly sales in selected stores of a retail store chain. 20 items that sold about 3 times a week were randomly collected and their sales in a period of four month were recorded. The task was to predict next weeks sales based on the sales of the past four weeks.

Kernel	MAE	MSE
dot	2.532	18.600
RBF, $\gamma = 0.1$	2.333	16.365
RBF, $\gamma = 1$	2.023	15.067
RBF, $\gamma = 10$	1.217	12.429
RBF, $\gamma = 100$	1.637	4.518
fourier, $q = 0.25$	2.851	24.137
fourier, $q = 0.5$	2.885	24.955
fourier, $q = 0.75$	2.975	24.767
subseq, $\gamma = 0.1$	2.621	20.221
subseq, $\gamma = 1$	2.650	20.761
subseq, $\gamma = 10$	2.749	22.140
subseq, $\gamma = 100$	2.788	22.874
PHMM	2.722	22.488

As can be seen, the RBF kernel with parameter $\gamma = 10$ shows the best results, followed by the RBF kernel with $\gamma = 100$. All other kernels show quite similar performance.

This is consistent with previous experiments with these time series. As only slow selling products were regarded, each week's sales can be very much attributed to random effects or effects that cannot be explained in terms of previous sales figures. Therefore, the time series models that were described in the previous section do not apply.

4.3 Intensive Care Data

This data consists of the minutely measurements of different vital signs of intensive care patients. These univariate time series have been classified by an experienced intensivist into three groups: Time series where a significant change in the level of the observations occurs (level change), time series with an outlier and time series without any change. A sequence of 20 minutes that contained the point of the change in the pattern of the time series have been extracted and used as the examples.

The learning task was to distinguish level changes against the other classes, resulting in 18 positive and 80 negative examples.

Kernel	Accuracy
dot	62.2%
RBF, $\gamma = 0.01$	73.3%
RBF, $\gamma = 1$	82.2%
RBF, $\gamma = 100$	81.1%
fourier, $q = 0.25$	81.1%
fourier, $q = 0.5$	81.1%
fourier, $q = 0.75$	78.8%
subseq, $\gamma = 1$	81.1%
PHMM	83.5%

The dot kernel completely fails to grasp the concept to be learned. All other kernels perform similar, with the PHMM kernel best.

The time series models that are based on sequences work well with this dataset because of the way the examples were generated: As the point of the pattern change could lie anywhere in the windows of the time series that was used as an examples, the matching of two time series requires to adjust both time indexes to each other.

5 Conclusions

The paper has presented different SVM kernels that can be used for univariate and multivariate time series analysis. Each of these kernels models different assumptions on the process that generates the time series. How to efficiently find out which kernel is optimal for a given learning task is still an unsolved problem.

The experiments showed that the RBF kernel performs very well on different types of time series and learning tasks. However, in specialized applications it may pay to have a close look on the time series model to be used.

Acknowledgments

The financial support of the Deutsche Forschungsgemeinschaft (SFB 475, "Reduction of Complexity for Multivariate Data Structures") is gratefully acknowledged.

References

1. Marcus Bauer, Ursula Gather, and Michael Imhoff. The identification of multiple outliers in online monitoring data. Technical Report 29/1999, Sonderforschungsbereich 475, Universität Dortmund, 1999.
2. Donald J. Berndt and James Clifford. Finding patterns in time series: A dynamic programming approach. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 3, pages 229–248. AAAI Press/The MIT Press, Menlo Park, California, 1996.
3. Alexander Clark. Learning morphology with pair hidden markov models. In *Proceedings of the Student Workshop at ACL 2001*, Toulouse, July 2001.
4. Gautam Das, Dimitrios Gunopulos, and Heikki Mannila. Finding similar time series. In *Principles of Data Mining and Knowledge Discovery*, pages 88–100, 1997.
5. Rodrigo Fernandez. Predicting time series with a local support vector regression machine. In *ACAI 99*, 1999.
6. David Haussler. Convolution kernels on discrete structures. Technical report, University of California at Santa Cruz, Department of Computer Science, Santa Cruz, CA 95064, USA, July 1999.
7. Huma Lodhi, John Shawe-Taylor, Nello Christianini, and Chris Watkins. Text classication using string kernels. *Advances in Neural Information Processing Systems*, 13, 2001.
8. Katharina Morik. The representation race – preprocessing for handling time phenomena. In Ramon Lopez de Mantaras and Enric Plaza, editors, *Machine Learning: ECML 2000*, Lecture Notes in Artificial Intelligence. Springer, 2000.
9. Sayan Mukherjee, Edgar Osuna, and Frederico Girosi. Nonlinear prediction of chaotic time series using support vector machines. In *Proc. of IEEE NNSP 97*, Sep 1997.
10. K. Müller, A. Smola, G. Ratsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In *Proceedings of the International Conference on Artificial Neural Networks*, Springer Lecture Notes in Computer Science. Springer, 1997.
11. L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.
12. Oliver Ritthoff, Ralf Klinkenberg, Simon Fischer, Ingo Mierswa, and Sven Felske. YALE: Yet another machine learning environment. In Ralf Klinkenberg, Stefan Rüping, Andreas Fick, Nicola Henze, Christian Herzog, Ralf Molitor, and Olaf Schröder, editors, *Proceedings of the LLWA 01 – Tagungsband der GI-Workshop-Woche* Lernen – Lehren – Wissen – Adaptivität, Forschungsberichte des Fachbereichs Informatik, Universität Dortmund, Dortmund, Germany, 2001.
13. Stefan Rüping. *mySVM-Manual*. Universität Dortmund, Lehrstuhl Informatik VIII, 2000. http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/.
14. Bernhard Schölkopf. The kernel trick for distances. Technical report, Microsoft Research, May 2000.
15. V. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, GB, 1998.
16. Chris Watkins. Dynamic alignment kernels. Technical report, Royal Holloway, University of London, 1999.