Long Term Prediction of Product Quality in a Glass Manufacturing Process Using a Kernel Based Approach

Tobias Jung[1], Luis Herrera[2], and Bernhard Schoelkopf[3]

[1] Johannes Gutenberg-Universitaet,
Fachbereich Mathematik & Informatik, 55099 Mainz, Germany
[2] University of Granada,
Dpt. of Computer Architecture and Technology, 18071 Granada, Spain
[3] M.P.I. for Biological Cybernetics, 72076 Tuebingen, Germany

Abstract In this paper we report the results obtained using a kernel-based approach to predict the temporal development of four response signals in the process control of a glass melting tank with 16 input parameters. The data set is a revised version[1] from the modelling challenge in EUNITE-2003. The central difficulties are: large time-delays between changes in the inputs and the outputs, large number of data, and a general lack of knowledge about the relevant variables that intervene in the process. The methodology proposed here comprises Support Vector Machines (SVM) and Regularization Networks (RN). We use the idea of sparse approximation both as a means of regularization and as a means of reducing the computational complexity. Furthermore, we will use an incremental approach to add new training examples to the kernel-based method and efficiently update the current solution. This allows us to use a sophisticated learning scheme, where we iterate between prediction and training, with good computational efficiency and satisfactory results.

1 Introduction

In this paper we report on using kernel based methods to predict the temporal development of four response signals that are quality-related process variables occurring in the manufacturing of glass. The data, provided by Schott Glass (Mainz), consists of 16 input values and four output values (each recorded at 15-minute intervals) and is the rescaled operational data of a glass melt obtained over a period of forty weeks. For the last two weeks, only the 16 input values are known, i.e. the control targets of the process engineers and the sometimes unexpected, but measurable external influences (like e.g. outside temperature), and our goal was to predict the four output values. The real physical meaning of the data and the manner in which the melting reacts to these inputs was not given. The central difficulty in forecasting the outputs is the seemingly irregular

[1] To obtain the data set cwt_2004, please contact katharina.lankers@schott.com

behavior with random spikes and the unknown delay time between variation of an input signal and the "response" from the glass melting tank.

Several paradigms and methodologies have been applied for time series prediction problems. Specifically, support vector machines (SVM) and kernel-based methods (KM) are receiving increasing attention [6,8], due to its remarkable characteristics such as good generalization performance, the absence of spurious local minima, the possibility of sparse representation of the solution and the relative independence of the computational complexity on the number of input dimensions of the problem [4,7]. Nevertheless, one disadvantage of KMs is that the computational complexity scales at least quadratically in the number of training samples. Thus a large amount of computation time will be involved when KMs are applied for solving large-size problems.

In this paper we present a kernel based approach that makes use of the idea of the sparse matrix approximation [3] in order to reduce the computational complexity of the problem and also as a means of regularization. Additionally, the performance of the model is improved using an iterative process that augments the original training sequence, by increasingly adding as new input variables noisy copies of the original data (obtained by learned 1-step predictions). The results obtained for the long term prediction problem showed that our approach obtained satisfactory results in comparison to those presented for the EUNITE 2003 competition [1].

The rest of the paper is organized as follows. Section 2 presents the kernel based learning methodology used in this paper. Section 3 presents the application of the methodology to the `cwt_2004` data set and the predictions obtained. Section 4 concludes the work.

2 Methodology, Kernel based Learning

2.1 SVM and RN

Given a data set of ℓ examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{\ell}$ with $\boldsymbol{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ being the inputs and $y_i \in \mathcal{Y} \subset \mathbb{R}$ being the outputs, the goal is to learn the underlying model. In this paper, we will consider as the space of candidate functions, the Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathcal{Y}$ endowed with reproducing kernel k, where $k : \mathcal{X} \times \mathcal{X} \to \mathcal{Y}$ is a symmetric, positive definite function (e.g. think of Gaussian RBF). The underlying function can be thus reconstructed solving a Tikhonov functional of the following general form

$$\min_{f \in \mathcal{H}} H[f] = \frac{1}{\ell} \sum_{i=1}^{\ell} c(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) + \Lambda \, \|f\|_{\mathcal{H}}^2 \tag{1}$$

where the first term measures the error in the approximation and the second term measures the complexity (i.e. the smoothness) of the current candidate. The Representer Theorem tells us that any solution to (1) has a representation in the form: $f(\cdot) = \sum_{i=1}^{\ell} \beta_i k(\mathbf{x}_i, \cdot)$ (i.e. as a sum of kernels centred on the data) where β_i are the coefficients that need to be determined [4]. The choice of the cost function c in (1) leads to two different methodologies:

– Quadratic costs $c(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) = \frac{1}{2}(y_i - f(\mathbf{x}_i))^2$ lead to RN. The coefficients β_i are obtained by solving the linear system

$$\boldsymbol{\beta} = (K^T K + \ell \Lambda K)^{-1} K^T \boldsymbol{y} = (K + \ell \Lambda I)^{-1} \boldsymbol{y}$$

with symmetric kernel matrix K. Note that we have to estimate one coefficient for each data point.

– ε-insensitive costs $c(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) = \max(0, |y_i - f(\mathbf{x}_i)| - \varepsilon)$ lead to Support Vector Regression. The coefficients β_i are obtained using $\beta_i = (\alpha_i^* - \alpha_i)$ and solving the constrained quadractic programming (QP)

$$\min_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^* \in \mathbb{R}^\ell} -\frac{1}{2}(\boldsymbol{\alpha}^* - \boldsymbol{\alpha})^T K (\boldsymbol{\alpha}^* - \boldsymbol{\alpha}) - \varepsilon(\boldsymbol{\alpha}^* + \boldsymbol{\alpha})^T \mathbf{e} + (\boldsymbol{\alpha}^* - \boldsymbol{\alpha})^T \mathbf{y} \qquad (2)$$

subject to $\mathbf{0} \leq \boldsymbol{\alpha}^*\ , \boldsymbol{\alpha} \leq C\boldsymbol{e}.$, where $\boldsymbol{\alpha}, \boldsymbol{\alpha}^* \in \mathbb{R}^\ell$ denotes the unknowns, $K \in \mathbb{R}^{\ell \times \ell}$ the kernel matrix, $\boldsymbol{e} = (1, \ldots, 1)^T$ and $C \in \mathbb{R}_{\geq 0}$ corresponds to the regularization parameter. Note that we have to calculate as many coefficients as we have data.

From a practical point of view, in order to apply these methods we need to choose (e.g. via cross validation) the kernel k, the tolerance ε and the right amount of regularization C or Λ.

In both cases the overall computational demand increases (at least) quadratically in the number of data. Moreover, if we were to incrementally add new samples and simultaneously needed to predict using the currently available model, we would constantly have to retrain our solution from scratch. Clearly this is completely infeasible. The next subsection describes a possible solution to this problem.

2.2 Sparse Approximation and incremental learning

The technique of *sparse approximation* allows us to dramatically reduce the number of variables considered in the optimization problem. It is based on the observation that the kernel matrix K often has rapidly decaying eigenvalues, and thus that the data in the feature space spans a rather low-dimensional manifold. Instead of using all the data, we can restrict us to use only the data points that compose the basis of this manifold. Therefore, we can approximate the data set by choosing just a few samples (a suitable basis) and projecting the remaining ones onto their span [2,3,4,5] (this technique is very common in the context of Gaussian process regression).

For example let the first m samples be our basis $\{k(\mathbf{x}_i, \cdot)\}_{i=1}^m$ ($m \ll \ell$). Then the remaining $\ell - m$ ones can be approximated via linear combination

$$k(\mathbf{x}_i, \cdot) \approx \sum_{j=1}^{m} a_{ij} k(\mathbf{x}_j, \cdot) \qquad , \ i = m+1 \ldots \ell. \qquad (3)$$

Since we are dealing with elements in RKHS $\mathcal{H}$, we have to determine coefficients a_{ij} such that the distance

$$d_i := \left\| k(\mathbf{x}_i, \cdot) - \sum_{j=1}^{m} a_{ij} k(\mathbf{x}_j, \cdot) \right\|_{\mathcal{H}}^2 \quad , \; i = m+1 \dots \ell \tag{4}$$

is minimized. Writing (4) in terms of the inner product (and recalling that in $\mathcal{H}$ we have $\langle k(\mathbf{x}_i, \cdot), k(\mathbf{x}_j, \cdot)\rangle_{\mathcal{H}} = k(\mathbf{x}_i, \mathbf{x}_j)$) and setting its derivate to zero leads to

$$\mathbf{a}_i = \tilde{K}^{-1}\tilde{\mathbf{k}}_i \tag{5}$$

where $\tilde{K} \in \mathbb{R}^{m\times m}$ with $[\tilde{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $\tilde{\mathbf{k}}_i \in \mathbb{R}^m$ with $\tilde{\mathbf{k}}_i = \big(k(\mathbf{x}_1, \mathbf{x}_i), \dots, k(\mathbf{x}_m, \mathbf{x}_i)\big)^T$. Once we have determined $\mathbf{a}_i$ corresponding to input $\mathbf{x}_i$, we obtain the corresponding approximation error as

$$d_i = k_{ii} - \tilde{\mathbf{k}}_i^T \mathbf{a}_i \tag{6}$$

This quantity tells us how well the basis $\{k(\mathbf{x}_i, \cdot)\}_{i=1}^m$ is able to approximate a given sample. Now it is very straightforward to use this error as a guide to (greedily) build up the basis online [9], thus allowing incremental addition of data: every time a new sample arrives, we check whether it can be approximated by the current basis well enough. If the error is below some chosen threshold, we do not need to add the current sample to the basis and can hence discard it. Only samples that cannot be approximated well are added to the basis. This incremental approach for building the manifold basis usually reduces the total number of samples to a small fraction of it.

2.3 Solving a reduced Problem

Thus, we can use the sparse greedy approximation to reduce the computational workload [3]: instead of solving a QP in ℓ variables, we only need to consider the m variables corresponding to the selected basis (usually $m \ll \ell$). Let $A \in \mathbb{R}^{\ell\times m}$ be the matrix consisting of rows $\mathbf{a}_i$ from (5). Then we approximate the full kernel matrix $K \in \mathbb{R}^{\ell\times\ell}$ via $K \approx A\tilde{K}A^T$. We define *reduced* variables $\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\alpha}}^* \in \mathbb{R}^m$ by setting $\tilde{\boldsymbol{\alpha}} = A^T\boldsymbol{\alpha}$ and $\tilde{\boldsymbol{\alpha}}^* = A^T\boldsymbol{\alpha}^*$. Now instead of solving the full QP (2) we can formulate a *reduced* QP: we replace K by $A\tilde{K}A^T$ and $\mathbf{y}$ by $\tilde{\mathbf{y}} = A^\dagger\mathbf{y}$ and obtain

$$\min_{\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\alpha}}^* \in \mathbb{R}^m} -\frac{1}{2}(\tilde{\boldsymbol{\alpha}}^* - \tilde{\boldsymbol{\alpha}})^T \tilde{K} (\tilde{\boldsymbol{\alpha}}^* - \tilde{\boldsymbol{\alpha}}) - \varepsilon(\tilde{\boldsymbol{\alpha}}^* + \tilde{\boldsymbol{\alpha}})^T A^\dagger \mathbf{e} + (\tilde{\boldsymbol{\alpha}}^* - \tilde{\boldsymbol{\alpha}})^T \tilde{\mathbf{y}} \tag{7}$$

Note that if we solve the *reduced* problem (7), we obtain the same result as if we had solved the full problem (2) using only the training samples $\{\mathbf{x}_i, \tilde{\mathbf{y}}_i\}_{i=1}^m$ in the basis. However, solving the reduced problem only depends on m and is asymptotically independent of ℓ. A similar reduction to an m-by-m problem is obtained in the case of regularization networks [5].

3 Predicting the Schott data `cwt_2004`

The data set `cwt_2004` consists of 20 measurements obtained in regular intervals (once every 15 minutes) from a glass melting process. Each row contains 20 observations from one time step. The first 16 observations (denoted by `in1`–`in16`) are measurements that *could* have an influence on the targets (like e.g. room temperature). The remaining 4 observations `out1`–`out4` are the targets that we want to predict. The task in the EUNITE-2003 challenge was the following: use the given values for time steps 1-16 000 to learn a model that predicts `out1`–`out4` for the next 1568 steps.

3.1 Preprocessing

The given data set `cwt_2004` consisted of raw measurements. We used the following steps during preprocessing of the data: replace missing values by an estimate (linear interpolation), remove outliers, apply data smoothing (rolling mean), normalize inputs and de-trend (remove increasing trend in first output, that often can not be implicitly considered by traditional learning methodologies).

3.2 Timeseries prediction as supervised learning

To obtain a model suitable for future predictions we use the supervised learning framework with 1-step predictions as targets and λ past observations as inputs. Thus our training samples $\{(\mathbf{x}_t, y_t)\}_t$ were given by

$$\mathbf{x}_t = \big(\texttt{out}_{t-1}, \ldots, \texttt{out}_{t-\lambda},\ \ \texttt{inp1}_t, \ldots, \texttt{inp1}_{t-\lambda}\ , \ldots,\ \texttt{inp16}_t, \ldots, \texttt{inp16}_{t-\lambda}\big)^T$$
$$y_t = \texttt{out}_t$$

We used lag $\lambda = 40$. Each of the 4 outputs is treated independently, that is we train 4 different models. Instead of using the usual approach of feeding the training examples *once* to the function approximator, we used a more complicated, iterated training procedure adapted from [2] (dropping the `inp` to avoid cluttered notation):

1. Iteration: using $\{(\mathbf{x}_i, y_i)\}_{i=\lambda}^{\ell}$ we obtain the model f^1. With f^1 we can compute the 1-step predictions for the *known* time steps $i = \lambda + 1 \ldots \ell$ to obtain $\hat{y}_i^1 = f^1(\mathbf{x}_i)$. Now we assemble the modified training samples $\{(\mathbf{x}_i^1, y_i)\}_{i=\lambda}^{\ell}$ with
$$\mathbf{x}_i^1 = \big(\hat{y}_{i-1}^1, y_{i-2}, y_{i-3}, \ldots, y_{i-\lambda},\ \ldots\big)^T$$
and add them to the current training set.
2. Iteration: using $\{(\mathbf{x}_i, y_i)\}_{i=\lambda}^{\ell} \cup \{(\mathbf{x}_i^1, y_i)\}_{i=\lambda}^{\ell}$ we obtain model f^2. With f^2 we can compute the 1-step predictions for the *known* time steps $i = \lambda + 1 \ldots \ell$ to obtain $\hat{y}_i^2 = f^2(\mathbf{x}_i)$. Now we assemble the modified training samples $\{(\mathbf{x}_i^2, y_i)\}_{i=\lambda}^{\ell}$ with
$$\mathbf{x}_i^2 = \big(\hat{y}_{i-1}^2, \hat{y}_{i-2}^2, y_{i-3}, \ldots, y_{i-\lambda},\ \ldots\big)^T$$
and add them to the current training set.

3. Iteration etc.

Thus, in the kth iteration we build our model using the training sequence

$$\{(\mathbf{x}_i, y_i)\}_{i=\lambda}^{\ell} \cup \{(\mathbf{x}_i^1, y_i)\}_{i=\lambda}^{\ell} \cup \ldots \cup \{(\mathbf{x}_i^k, y_i)\}_{i=\lambda}^{\ell} \; .$$

Note that this iterated training procedure greatly benefits from *sparse approximation* which drastically reduces the computational complexity. In the simulations performed, using repeated training resulted in much better predictions when compared with the results obtained using just one iteration. Thus we could observe that augmenting the original training sequence by increasingly "noisy" copies of the original data (where the "noise" is obtained by the learned 1-step predictions) is very helpful in getting good forecasts.

3.3 Results

In this section we compare the resulting predictions obtained from RN and SVM using *sparse approximation* in both cases and Gaussian RBF kernels. To determine the parameters, we used the following four validation sets: (1) train 1-13 500 predict 13 501-14 500 (2) train 1-14 000 predict 14 001-15 000 (3) train 1-14 500 predict 14 501-15 500 (4) train 1-15 000 predict 15 001-16 000. The optimal parameters that led to the best (averaged) prediction error during validation were then used to obtain the final model. These parameters are: (output1) `TOL=0.01`, $\sigma = 0.015$ (output2) `TOL=0.011`, $\sigma = 0.005$ (output3) `TOL=0.01`, $\sigma = 0.013$ (output3) `TOL=0.009`, $\sigma = 0.013$. The parameters governing regularization ($C = 500, \Lambda = 0.1$) and tolerance ($\varepsilon = 0.1$) were determined outside the validation loop.

Forecasting the global trend of the outputs is more important than the modelling of high-frequency variations. To this end a custom error measure TubeERR was devised, that does not penalize small deviations. Also an emphasize is done in that forecasts into the far future are less reliable. Thus the TubeErr measure is defined as

$$\text{TubeERR}(i) = \max\big(0, |y_i - f(\mathbf{x}_i)| - (3 + 0.003 * i)\big) \qquad , i = 0, 1, \ldots 1568 - 1$$

Thus, to evaluate the quality of the predictions, we use MSE and the TubeERR error measure provided by Schott. Using these two measures, the optimal number of iterations in the iterated learning scheme was calculated using pre-tests on the validation sets. The achieved error is summarized in Table 1. Figure 1 shows the resulting predictions using SVM; see how for the four outputs, the prediction obtained follows the main variations and is inside the TubeERR region most of the 1568 time steps.

4 Conclusions

In this paper we have reported the use of a kernel-based method to predict the long term time series used in the data modelling competition EUNITE-2003. We

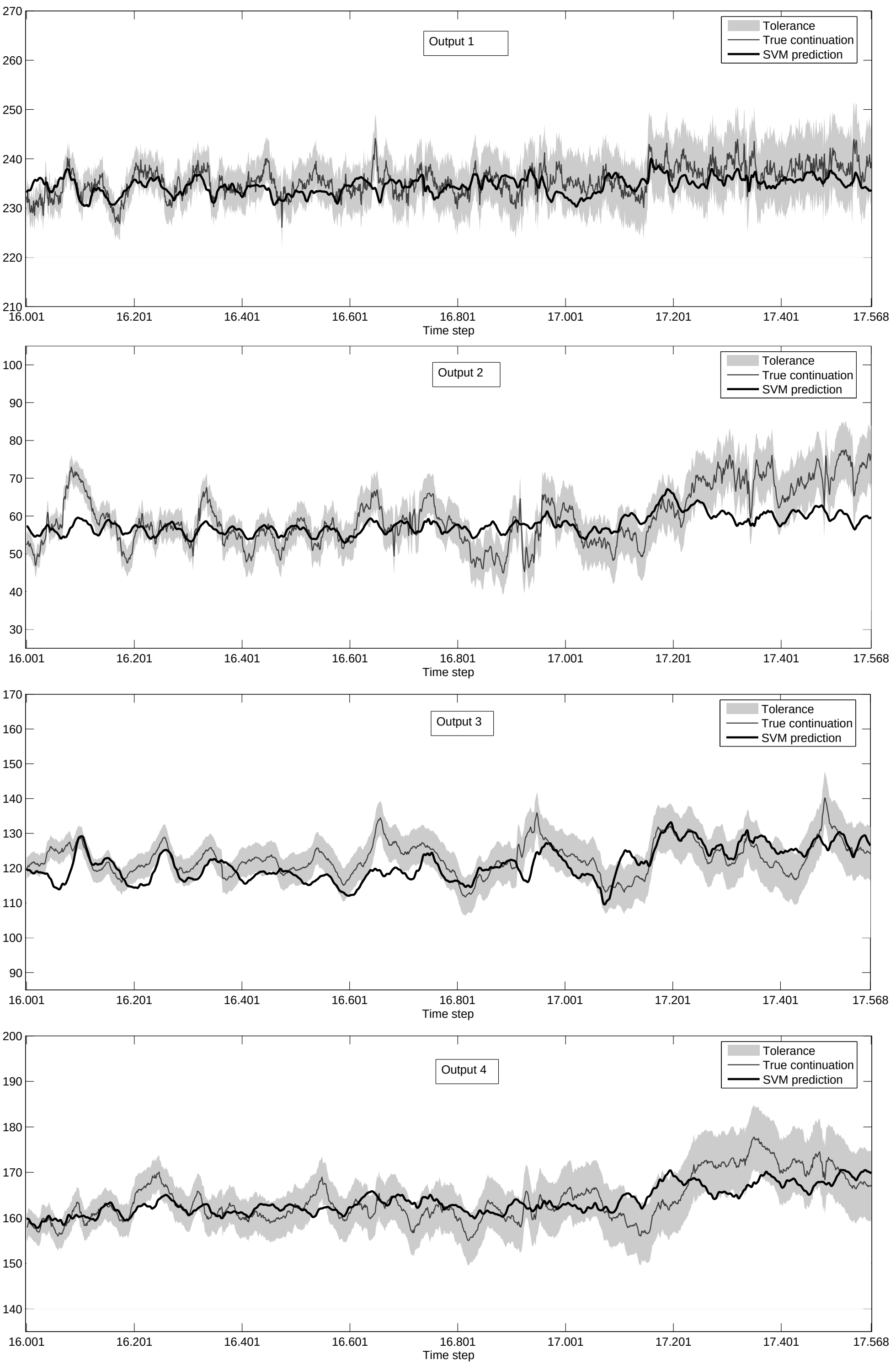

Figure 1. Results using SVMs. We show the original outputs and the prediction obtained using SVMs. The gray curves mark the boundaries of the tolerated error TubeERR.

Table 1. Comparison of the prediction errors for time step 16 001-17 568.

		SVM	**RN**
output 1	MSE	7.44	9.44
TOL=0.01	TubeERR	115.54	149.87
$\sigma = 0.015$	#Basis	591	
output 2	MSE	41.83	47.41
TOL=0.011	TubeERR	2082.12	2382.44
$\sigma = 0.005$	#Basis	195	
output 3	MSE	24.07	22.28
TOL=0.01	TubeERR	1214.25	1069.10
$\sigma = 0.013$	#Basis	476	
output 4	MSE	20.84	23.29
TOL=0.009	TubeERR	781.02	1115.24
$\sigma = 0.013$	#Basis	486	

have used the idea of sparse approximation as a means of reducing the computational complexity and an incremental learning approach to add new training samples to the kernel-based method and efficiently update the current solution. The results obtained support the suitability of the proposed methodology for this long term time series prediction problem with high training complexity.

References

1. EUNITE Competition 2003: Prediction of product quality in glass manufacturing, www.eunite.org (2003)
2. Engel, Y., Mannor, S., Meir, R.: The Kernel Recursive Least Square Algorithm. IEEE Transactions on Signal Processing, **52(8)** (2004) 2275–2285
3. Engel, Y., Mannor, S., Meir, R.: Sparse online greedy support vector regression. Proc. of 13th European Conference on Machine Learning. Springer (2002)
4. Schoelkopf, B., Smola, A.: Learning with Kernels. Cambridge, MA: MIT Press (2002)
5. Smola, A., Schoelkopf, B.: Sparse greedy matrix approximation for machine learning. Proc. of 17th International Conference on Machine Learning. Morgan Kaufmann (2000)
6. Cao, L.J., Tay, F.E.H.: Support Vector Machine With Adaptive Parameters in Financial Time Series Forecasting. IEEE Transactions on Neural Networks. **14(6)** (2003) 1506-1518
7. Cristianini, N, Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Preess. Cambridge, England (2000)
8. Chang, M.-W., Chen, B.-J., Lin, C.-J.: EUNITE Network Competition: Electricity Load Forecasting , November 2001. Winner of EUNITE world wide competition on electricity load prediction.
9. Csato, L., Opper,M.: Sparse on-line Gaussian processes. Neural Computation **14(3)** (2002) 641–669