Long-Term Prediction of Time Series Using State-Space Models

Elia Liitiäinen and Amaury Lendasse

Neural Networks Research Centre, Helsinki University of Technology, Finland
eliitiai@cc.hut.fi, lendasse@hut.fi

Abstract. State-space models offer a powerful modelling tool for time series prediction. However, as most algorithms are not optimized for long-term prediction, it may be hard to achieve good prediction results. In this paper, we investigate Gaussian linear regression filters for parameter estimation in state-space models and we propose new long-term prediction strategies. Experiments using the EM-algorithm for training of nonlinear state-space models show that significant improvements are possible with no additional computational cost.

1 Introduction

Time series prediction [1] is an important problem in many fields like ecology and finance. The goal is to model the underlying system that produced the observations and use this model for prediction.

Often, obtaining prior information of a time series is difficult. In this kind of a situation, a black-box model can be used. An often used approach for prediction is nonlinear regression, which works well especially for deterministic time series.

An alternative to nonlinear regression is state-space modeling. A state-space model can model a wide variety of phenomena, but unfortunately learning a state-space presentation for data is challenging. Previous work on this topic includes dual Kalman filtering methods [2], variational Bayesian learning [3] and EM-algorithm for nonlinear state-space models [4].

In [4] an EM-algorithm for training of state-space models is proposed. From the point of view of time series prediction, the drawback of this algorithm is that the model is not exactly optimized for prediction.

In this paper, we have two goals. It is shown that the EKS which was used in [4] can be replaced by a more efficient linear regression smoother. We also show that with simple methods it is possible to improve the long-term prediction ability of the algorithm. The improvement comes with no additional computational cost and our methods can be applied also to other algorithms. In the experimental section, we test the proposed methods with the Poland electricity and Darwin sea level pressure datasets.

S. Kollias et al. (Eds.): ICANN 2006, Part II, LNCS 4132, pp. 181–190, 2006.

2 Gaussian Linear Regression Filters

Consider the nonlinear state-space model

$$x_k = f(x_{k-1}) + w_k \tag{1}$$
$$y_k = h(x_k) + v_k, \tag{2}$$

where $w_k \sim N(0, Q)$ and $v_k \sim N(0, r)$ are independent Gaussian random variables, $x_k \in R^n$ and $y_k \in R$. Here, $N(0, Q)$ means normal distribution with the covariance matrix Q. Correspondingly, r is the variance of the observation noise.

The filtering problem is stated as calculating $p(x_k|y_1, \ldots, y_k)$. If f and h are linear, this could be done using Kalman filter [5]. The linear filtering theory can be applied to nonlinear problems by using Gaussian linear regression filters (LRKF, [6]) that are recursive algorithms based on statistical linearization of the model equations. The linearization can be done in various ways resulting in slightly different algorithms.

Denote by $\tilde{p}(x_k|y_1, \ldots, y_k)$ a Gaussian approximation of $p(x_k|y_1, \ldots, y_k)$. The first phase in a recursive step of the algorithm is calculating $\tilde{p}(x_{k+1}|y_1, \ldots, y_k)$, which can be done by linearizing $f(x_k) \approx A_k x_k + b_k$ so that the error

$$\mathrm{tr}(e_k) = \mathrm{tr}\ (\int_{R^{n_x}} (f(x_k) - A_k x_k - b_k)(f(x_k) - A_k x_k - b_k)^T \tilde{p}(x_k|y_0, \ldots, y_k)\ dx_k) \tag{3}$$

is minimized. Here, tr denotes the sum of the diagonal elements of a matrix. In addition Q is replaced by $\tilde{Q}_k = Q + e_k$. The linearized model is used for calculating $\tilde{p}(x_{k+1}|y_1, \ldots, y_k)$ using the theory of linear filters. The measurement update

$$\tilde{p}(x_{k+1}|y_1, \ldots, y_k) \rightarrow \tilde{p}(x_{k+1}|y_1, \ldots, y_{k+1})$$

is done by a similar linearization. The update equations for the Gaussian approximations can be found in [7].

Approximating equation 3 using central differences would lead to the central difference filter (CFD, [5]) which is related to the unscented Kalman filter [5]. A first order approximation on the other hand would yield the widely used extended Kalman filter algorithm [8]. However, by using Gaussian nonlinearities as described in the following sections, no numerical integration is needed in the linearization. The filter based on closed form calculations is called a (Gaussian) linear regression filter.

The smoothed density $p(x_l|y_1, \ldots, y_k)$ $(l < k)$ is also of interest. In our experiments we use the Rauch-Tung-Striebel smoother [8] with the linearized model. Other alternatives are of course also of interested and will be possibly investigated in future research. See for example [9] in which three different smoothing methods are compared.

3 EM-Algorithm for Training of Radial Basis Function Networks

In this section, a training algorithm introduced in [4] is described. The EM-algorithm is used for parameter estimation for nonlinear state-space models. When Gaussian filters are used, the M-step of the algorithm can be solved in closed form resulting in an efficient method which in practice converges fast.

3.1 Parameter Estimation

Suppose that a sequence of observations $(y_k)_{k=1}^N$ is available. The underlying system that produced the observations is modelled as a state-space model. The functions f and g in equations 1 and 2 are parametrized using RBF-networks:

$$f = w_f^T \Phi_f \tag{4}$$

$$g = w_g^T \Phi_g, \tag{5}$$

where $\Phi_f = [\rho_1^f(x), \ldots, \rho_l^f(x)\ x^T\ 1]^T$ and $\Phi_g = [\rho_1^g(x), \ldots, \rho_j^g(x)\ x^T\ 1]^T$ are the neurons of the RBF-networks. The nonlinear neurons are of the form

$$\rho(x) = |2\pi S|^{-1/2} \exp(-\frac{1}{2}(x-c)^T S^{-1}(x-c)). \tag{6}$$

The free parameters of the model are the weights of the neurons, w_f and w_g in equations 4 and 5, and the noise covariances Q and r in equations 1 and 2. In addition, the initial condition for the states is chosen Gaussian and the parameters of this distribution are optimized.

The EM-algorithm is a standard method for handling missing data. Denoting by θ the free parameters of the model, the EM-algorithm is used for maximizing $p(y_1, \ldots, y_T|\theta)$. The EM-algorithm for learning nonlinear state-space models is derived in [4].

The algorithm is recursive and each iteration consists of two steps. In the E-step, the density $p(x_0, \ldots, x_N|y_1, \ldots, y_N, \theta)$ is approximated and in the M-step this approximation is used for updating the parameters.

Due to the linearity of the model with respect to the parameters, the M-step can be solved analytically. The update formulas for the weights w_f and w_g and covariances Q and R can be found in [4]. In our implementation Q is chosen diagonal.

The E-step is more difficult and an approximative method must be used. In addition to Gaussian filters, there exists many different methods for nonlinear smoothing, for example the particle smoother and numerical quadrature based methods [10]. In the case of neural networks, the problem is that function evaluation may be relatively expensive if a high number of neurons is used. We propose the use of the smoother derived in section 2 instead of the extended Kalman smoother used in [4]. The extended Kalman smoother uses rough approximations to propagate nonlinearities which may cause inaccuracy (see for example [11]). Our choice does not significantly increase the computational complexity of the algorithm which did not pose problems in our experiments.

3.2 Initialization

The state variables must be initialized to some meaningful values before the algorithm can be used. Consider a scalar time series (y_t). First the vectors

$$z_t = [y_{t-L}, \ldots, y_t, \ldots, y_{t+L}] \tag{7}$$

are formed. L is chosen large enough so that the vectors contain enough information. Based on Taken's theorem [12], it can be claimed that setting L to the dimension of the state-space is a good choice [3], but as many time series contain noise, this is not always the case. In our experiments, we use the value $L = 10$ which produced good results in our experiments and is large enough for most time series. In case of bad results, a lower value for L might sometimes give better initializations.

Next the dimension for the hidden states is chosen. Once the dimension is known, the vectors z_t are projected onto this lower dimensional space. This is done with the PCA mapping [13]. In highly nonlinear problems, it may be essential to use kernel PCA like was done in [14].

The rough estimates for the hidden states are used to obtain an initial guess for the parameters of the network.

3.3 Choosing Kernel Means and Widths

Choosing the centers of the neurons (c in equation 6) is done with the k-means algorithm [13]. The widths S_j are chosen according to the formula (see [15])

$$S_j = \frac{1}{l}(\sum_{i=1}^{l} \|c_j - c_{N(j,i)}\|^2)^{\frac{1}{2}} I, \tag{8}$$

where I is the identity matrix and $N(j, i)$ is the ith nearest neighbor of j. In the experiments, we use the value $l = 2$ as proposed in [15].

3.4 Choosing the Number of Neurons and the Dimension of the State-Space

To estimate the dimension of the state-space, we propose the use of a validation set to estimate the generialization error. For each dimension, the model is calculated for different number of neurons and the one which gives the lowest one-step prediction validation error is chosen. The linear regression filter is used for calculating the error.

For choosing the number of neurons, we propose the use of the likelihood of the model, which has the advantage that the whole data set is used for training. Usually there is a bend after which the likelihood decreases only slowly (see figure 1c). This kind of a bend is used as an estimate of the point after which overfitting becomes a problem.

4 Long-Term Prediction of Time Series

By long-term prediction of a time series we mean predicting y_{t+k} for $k > 1$ given the previous observations $y_1, \ldots, y_t$. The model in equations 1 and 2 can be used for prediction by calculating a Gaussian approximation to $p(y_{t+k}|y_1, \ldots, y_t)$ with the linear regression filter or the EKF. However, the drawback of this approach is that the EM-algorithm does not optimize the long-term prediction performance of the model and thus the prediction error is expected to grow fast as the prediction horizon k grows. Our claim is motivated for example by the results in [16], which show that for nonlinear regression direct prediction instead of recursive gives much better results on long-term prediction.

Instead of the previous method, we propose three alternative methods. For the first two methods the observation model in equation 5 is replaced by an RBF-network with the same centers and widths for the neurons than the original model but different weights that are optimized for long-term prediction. In the first method, the weights are optimized to minimize the prediction error when the linear regression filter with the original model is used to estimate $p(x_t|y_1, \ldots, y_t)$ and the EKF with the modified model for estimating $p(x_{t+k}|y_1, \ldots, y_t)$ and $p(y_{t+k}|y_1, \ldots, y_t)$.

In the second method, a similar optimization is performed in the case where the linear regression filter is used for estimating all three distributions.

As a third methods, we propose extending the first method so that also the weights of the network in equation 4 are reoptimized. This leads to a nonlinear optimization problem, which can be solved using standard methods, see for example [17].

The optimization of the weights for the methods is done using the same training set as for the EM-algorithm. The cost function is

$$\sum_{i=1}^{N-k} (y_{i+k} - \hat{y}_{i+k})^2,$$

where $\hat{y}_{i+k}$ are the predictions given the parameter values and observations up to time i. The three different methods are used for calculating $\hat{y}_{i+k}$. The first two lead to quadratic optimization problems which are easy to solve in closed form, whereas to optimize the cost for the third method any nonlinear optimization method can be used.

There certainly exists different methods than the ones we use. Our goal is not to test all different possibilities, only to show that when state-space models are used, an ordinary recursive prediction is not the best choice.

5 Experiments

In this section the proposed methods are tested on two different time series. The first time series represents the electricity consumption in Poland and is

interesting also from practical point of view as electricity companies can certainly make use of this kind of information. The other time series, Darwin sea level pressure, is also measurements of a real world phenomena. The data sets can downloaded from [18]. Unless we state explicitly otherwise, the linear regression filter based algorithm is used for training.

5.1 Poland Electricity Consumption

The Poland electricity consumption time series is a well-known benchmark problem [19]. For prediction results with nonlinear regression we refer to [16]. The series is plotted in figure 1a. The values $1 - 1000$ are used for training and the values $1001 - 1400$ for testing. For dimension selection, the values 601-1000 are kept for validation.

The model is tested using the dimensions 2 to 8 for the state-space. For each dimension, the validation errors are calculated for different number of neurons so that the number of neurons goes from 0 to 36 by step of 2 (we use the same amount of neurons for both networks). For each number of neurons, the training is stopped if the likelihood decreased twice in row or more than 40 iterations have been made.

The dimension 7 yields the lowest validation error and was thus chosen as the dimension for the states. In figure 1c is the corresponding likelihood curve for this dimension. Based on the likelihood, we choose 8 as the number of neurons.

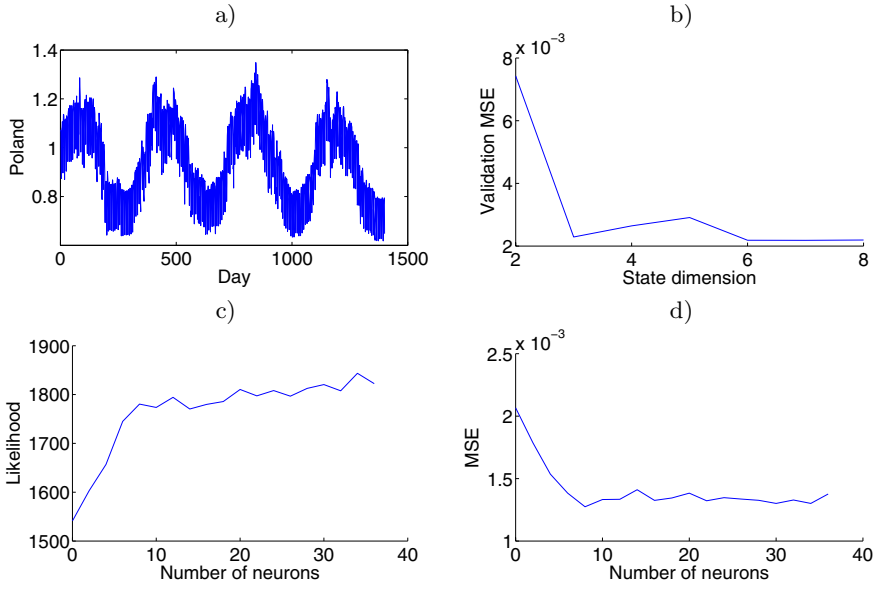

Fig. 1. a) The Poland electricity consumption time series. b) Validation errors for state dimensions 2-8. c) Likelihoods for dimension 7. d) One step prediction test errors for dimension 7.

From figure 1d, it can be seen that the short-term prediction performance of the model is good.

The chosen model is used for long-term prediction with different prediction horizons. We test the three methods introduced in section 4. The results are in figure 2a. In figure 2a, the test errors are calculated by estimating the state at each time point of the test set with the linear regression filter. After that the methods in section 4 are used to calculate the prediction estimates corresponding to the number of prediction steps.

In figure 2b for comparison we have implemented the EKS-based algorithm in [4], where for prediction the EKF has been used. Also for this algorithm we choose 7 as the dimension of the state-space and 8 as the number of neurons. Because the error for the algorithm in [4] has a quite high variance compared to our method, the curves in figure 2b are averaged over 5 simulations.

It can be seen that the linear regression filter in training improves the result compared to using EKS. Also, the stability of the algorithm is improved. The linear regression filter gives higher covariances for the smoothed state estimates which has a regularizing effect on the model.

For short-term prediction the prediction error is not significantly improved by the methods proposed in section 4. However, as the prediction horizon grows, the differences between the methods grow. Thus in this time series, the long-term prediction methods in section 4 should be used as they introduce no additional computational cost.

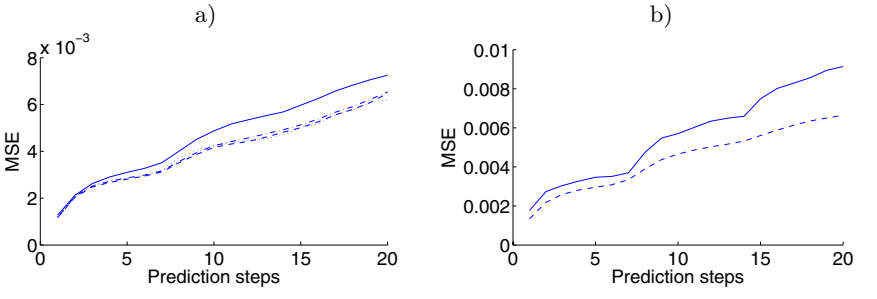

Fig. 2. Long-term prediction errors. a) The solid line is the recursive prediction error with the linear regression filter. The dashed, dashdotted and dotted lines give the prediction error for the other three methods in section 4 so that dashed corresponds to the first of these methods, dashdot to the second etc. b) Results with the original algorithm in [4] (solid line) and the basic version of our algorithm (dashed line). The curves in figure b are averaged over 5 simulations.

5.2 Darwin Sea Level Pressure

The Darwin sea level pressure time series consists of 1400 values which are drawn in figure 3a. The values from 1 to 1000 are used for training and the rest for testing the model.

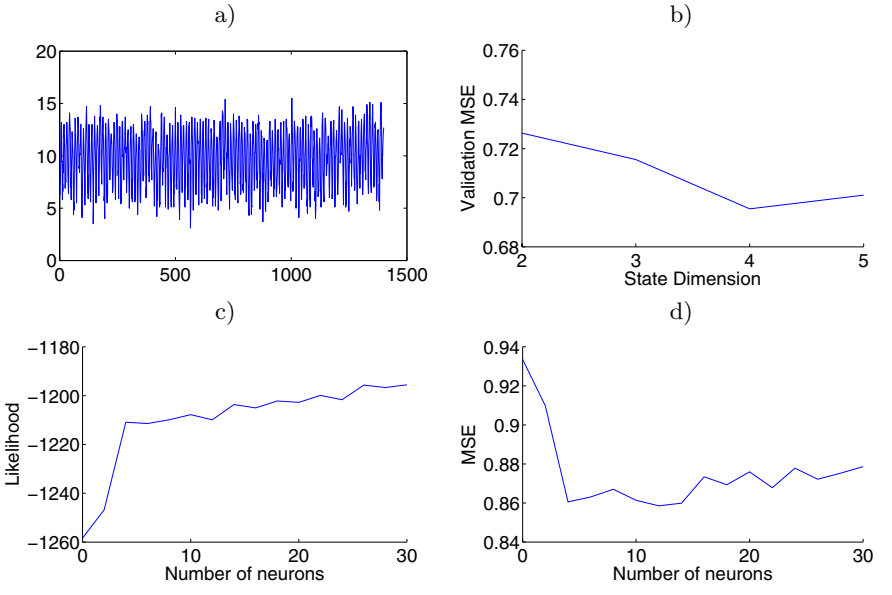

Fig. 3. a) The Darwin sea level pressure time series. b) Validation errors for state dimensions 2-5. c) Likelihoods for dimension 4. d) One step prediction test errors for dimension 4.

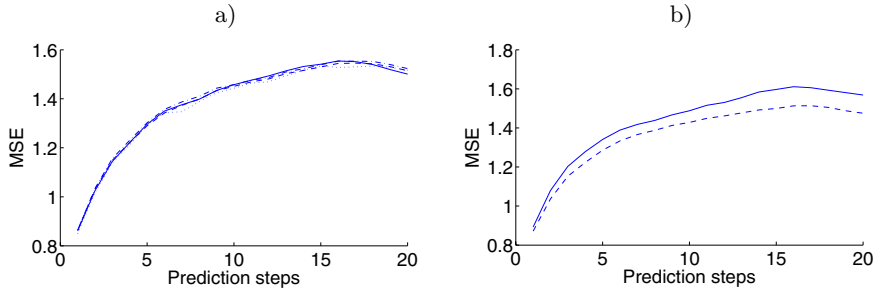

Fig. 4. a) The solid line is the recursive prediction error with the linear regression filter. The dashed, dashdotted and dotted lines give the prediction errors for the other three methods in section 4 so that dashed corresponds to the first of these methods, dashdot to the second etc. b) Results with the original algorithm in [4] (solid line) and the basic version of our algorithm (dashed line). The curves in figure b are averaged over 5 simulations.

As in the previous section, the dimension of the state-space is chosen using a validation set. The dimensions from 2 to 5 are tested. The lowest validation error is that of dimension 4. The corresponding likelihood curve is in figure 3c. There is a clear bend in the likelihood curve, which can be used for model selection. Based on the curve we choose 4 as the number of neurons.

From figure 4b, it can be seen that the original algorithm with EKS is again worse than our algorithm.

The long-term prediction results with the methods in section 4 are in figure 4a. In this problem, the difference between different prediction methods is much smaller. No significant improvement was obtained over recursive prediction. This is probably due to the linearity and periodicity of the data. In this time series, an accurate long-term prediction is possible.

Based on the experiments, we claim that the best of the methods proposed in section 4 is the one based on minimizing the prediction performance of the EKF. It seems that using Gaussian distributions instead of point values as a training set is not very useful.

6 Conclusion

In this paper the Gaussian linear regression smoother with the EM-algorithm is used as a method for learning a state-space presentation for a time series. It is shown that our approach brings real improvement over the original EKS based algorithm presented in [4] for significantly nonlinear problems. The proposed smoother gives implicitely additional regularization compared to the extended Kalman smoother. Still it is clear that the EM-algorithm does not minimize the long-term prediction error of the model which results in high prediction errors once the prediction horizon grows.

In this paper we proposed strategies for improving the long-term prediction capability of state-space models. The methods can be applied for many algorithms in addition to the EM-algorithm used in this paper. The experimental results show that clear improvement over the ordinary recursive approach is possible. However, for nearly linear time series the results were not as convincing and it can be concluded that big improvements can be obtained mainly for significantly nonlinear systems.

In the future methods for optimizing state-space models for long-term prediction will be investigated. Most current methods use a cost function that poorly fits to long-term prediction.

References

1. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley Publishing Company, 1994.
2. A. T. Nelson. *Nonlinear Estimation and Modeling of Noisy Time-series by Dual Kalman Filtering Methods*. PhD thesis, Oregon Graduate Institute, 2000.
3. H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 14(11):2647–2692, 2002.
4. Z. Ghahramani and S. Roweis. Learning nonlinear dynamical systems using an EM algorithm. In *Advances in Neural Information Processing Systems*, volume 11, pages 431–437. 1999.
5. R. van der Merwe. *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*. PhD thesis, Oregon Health & Science University.

6. T. Lefebvre, H. Bruyninckx, and J. De Schutter. Kalman filters for non-linear systems: a comparison of performance. *International Journal of Control*, 77(7):639–653, 2004.
7. K. Ito and K. Q. Xiong. Gaussian filters for nonlinear filtering problems. *IEEE Transactions on Automatic Control*, 45(5):910–927, 2000.
8. S. Haykin, editor. *Kalman Filtering and Neural Networks*. Wiley Series on Adaptive and Learning Systems for Signal Processing. John Wiley & Sons, Inc., 2001.
9. T. Raiko, M. Tornio, A. Honkela, and J. Karhunen. State inference in variational bayesian nonlinear state-space models. In *6th International Conference on Independent Component Analysis and Blind Source Separation, ICA 2006, Charleston, South Carolina, USA, March 5-8, 2006*.
10. A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.
11. Eric A. Wan, R. van der Merwe, and A. T. Nelson. Dual estimation and the unscented transformation. In *Advances in Neural Information Processing Systems*, 2000.
12. D. A. Rand and L. S. Young, editors. *Dynamical Systems and Turbulence*, volume 898 of *Lecture Notes in Mathematics*, chapter Detecting strange attractors in turbulence. Springer-Verlag, 1981.
13. S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd edition, 1998.
14. A. Honkela, S. Harmeling, L. Lundqvist, and H. Valpola. Using kernel PCA for initialisation of variational bayesian nonlinear blind source separation method. In *Proceedings of the Fifth International Conference Independent Component Analysis and Blind Signal Separation (ICA 2004), Granada, Spain.*, volume 3195 of *Lecture Notes in Computer Science*, pages 790–797. Springer-Verlag, 2004.
15. J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, pages 281–294, 1989.
16. Y. Ji, J. Hao, N. Reyhani, and A. Lendasse. Direct and recursive prediction of time series using mutual information selection. In *Computational Intelligence and Bioinspired Systems: 8th International Workshop on Artificial Neural Networks, IWANN 2005, Vilanova i la Geltrú, Barcelona, Spain, June 8-10,2005*, pages 1010–1017.
17. R. Fletcher. *Practical Methods of Optimization*, volume 1. John Wiley and Sons, 1980.
18. Available from www.cis.hut.fi/projects/tsp/download.
19. M. Cottrell, B. Girard, and P. Rousset. Forecasting of curves using classification. *Journal of Forecasting*, 17(5-6):429–439, 1998.