ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

EM-algorithm for Training of State-space Models with Application to Time Series Prediction

Elia Liitiäinen, Nima Reyhani and Amaury Lendasse

Helsinki University of Technology - Neural Networks Research Center
P.O. Box 5400, FI-02015 - Finland
{elia, lendasse, nreyhani}@cis.hut.fi

Abstract. In this paper, an improvement to the E step of the EM algorithm for nonlinear state-space models is presented. We also propose strategies for model structure selection when the EM-algorithm and state-space models are used for time series prediction. Experiments on the Poland electricity load time series show that the method gives good short-term predictions and can also be used for long-term prediction.

1 Introduction

Time series prediction is an important problem in many fields. Applications include finance, prediction of electricity load and ecology. Time series prediction is a problem of system identification.

Taken's theorem [1] ensures that if the dynamic of an underlying system is deterministic, a regression approach can be used for prediction. However, many real world phenomena are stochastic and more complicated methods might be necessary. One possibility is to use a state-space model for predicting the behavior of a time series. This kind of model is very general and is able to model many kinds of phenomena.

Choosing the structure and estimating the parameters of a state-space model can be done in many ways. One approach is to use the EM-algorithm and Radial Basis Functions (RBF)-networks [2]. The main problem is that the E-step of the algorithm is difficult due to nonlinearity of the model. In this paper, we apply a Gaussian filter which handles nonlinearities better than the ordinary Extended Kalman Filter (EKF) which is used in [2].

The problem of choosing the dimension of the state-space and the number of neurons is also investigated. This problem is important as too complicated a model can lead to numerical instability, high computational complexity and overfitting. We propose the use of training error curves for model structure selection. The experimental results show that the method works well in practice. We also show that the EM-algorithm maximizes the short-term prediction performance but is not optimal for long-term prediction.

2 Gaussian Linear Regression Filters

Consider the nonlinear state-space model

$$\begin{aligned} x_k &= f(x_{k-1}) + w_k && (1) \\ y_k &= g(x_k) + v_k, && (2) \end{aligned}$$

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. The linear filtering theory can be applied to nonlinear problems by using Gaussian linear regression filters (LRKF) 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 $\tilde{Q}_k = Q_k + e_k$. 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.

Approximating equation 3 using central differences would lead to the central difference filter (CFD) which is related to the unscented Kalman filter. However, by using Gaussian nonlinearities as described in the following sections, no numerical integration is needed in the linearization. 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 [4] with the linearized model.

3 Expectation Maximization (EM)-algorithm for Training of Radial Basis Function Networks

In this section, a training algorithm introduced in [2] is described. The EM-algorithm is used for parameter estimation for nonlinear state-space models.

3.1 Parameter Estimation

Suppose 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:

$$\begin{aligned} f &= w_f^T \Phi_f && (4) \\ g &= w_g^T \Phi_g, && (5) \end{aligned}$$

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

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

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 [2].

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 [2]. In our implementation Q is chosen diagonal.

The E-step is more difficult and an approximative method must be used. We propose the use of the smoother derived in section 2 instead of the extended Kalman smoother used in [2].

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

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

are formed. L is chosen large enough so that the vectors contain enough information.

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 [5].

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 [5]. The widths S_j are chosen according to the formula (see [6])

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

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

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 generiealization error. For each dimension, the model is calculated for different number of neurons and the one which gives the lowest validation error is chosen. This procedure is repeated to get averaged estimates which are used for dimension selection.

For choosing the number of neurons, we propose the use of the training error, which has the advantage that the whole data set is used for training. Usually there is a bend after which the training error 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. To avoid local minima, averaging is used.

Because iterative prediction is used, the one-step prediction error is used for model structure selection. Instead, it would also have been possible to use the likelihood.

4 Experiments

The experiments are made with a time series that represents the daily electricity consumption in Poland during 1400 days in the 90s [7], see figure 1a. The values $1 - 1000$ are used for training and the values $1001 - 1400$ for testing. For the 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). To calculate the prediction errors, the state at the beginning of each prediction is estimated with the linear-regression filter. The prediction is made by approximating the future states and observations with the linear-regression approach. For L in formula 7, we choose 10. The choice is based on the results in [8] which imply that the window size 10 contains enough information for prediction.

For each number of neurons, the training is stopped if the likelihood decreased twice in row or more than 40 iterations have been made. In figure 1b are the validation MSE curves calculated by averaging the values for each number of

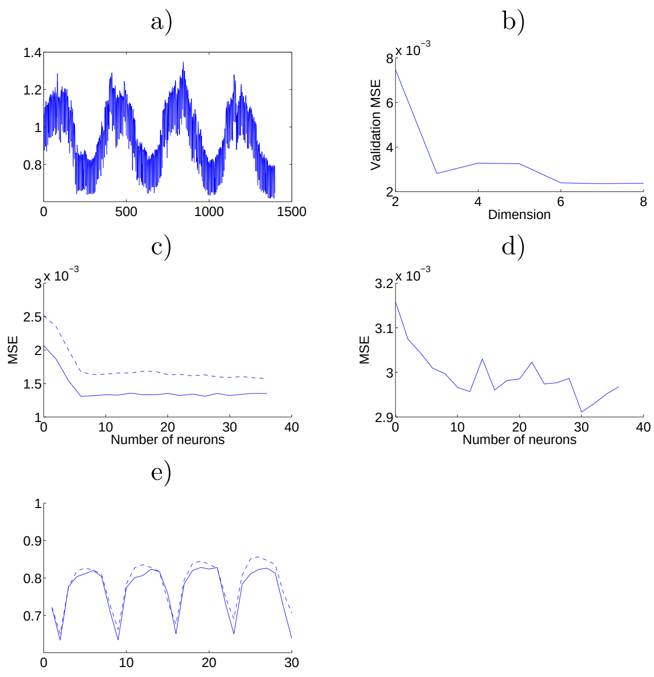

Fig. 1: a) The Poland electricity consumption time series. b) 1-step prediction validation errors for dimension selection. c) 1-step prediction errors for dimension 7. The solid line is the error on the test set and the dashed line the error on the training set. d) 5-step test error for dimension 7. e) An example of prediction.

neurons over four simulations and choosing the minimum for each dimension. It can be seen that a good choice for dimension of the state-space is 7 (even though the validation error for dimension 6 is nearly as good). The errors are averaged over four simulations.

The training error curve for 1-step prediction shows clearly that a good choice for the number of neurons is 6. The test error shows that the model chosen based on the training error gives good results on short-term prediction.

The chosen model gives quite good but not optimal 5-step prediction result. We claim that the EM-algorithm optimizes mainly the short-term prediction performance which can be seen by examining the test error curves. The 1-step test error curve is much smoother. Thus, an algorithm designed for long-term prediction would certainly give much better result.

In table 1 are the results corresponding to the averaged prediction over four simulations. In comparison we have calculated the corresponding results using linear regression, lazy learning [9] and LS-SVM [8]. As expected the 1-step MSE of the EM-algorithm is very good, but the 5-step MSE is less good.

method	1-step MSE	5-step MSE
EM	0.0013	0.0030
linear	0.0021	0.0038
lazy learning	0.0016	0.0026
LS-SVM	0.0015	0.0032

Table 1: Test errors with averaged predictions.

5 Conclusion

In this paper, a new approach to the E-step of the EM-algorithm for nonlinear state-space models is proposed. We also propose strategies for model structure selection. The experimental results show that the optimization of the model for short-term prediction works well but on the other hand the EM-algorithm is not optimal for long-term prediction. This is because the prediction is iterative with a model optimized for 1-step prediction.

The linear regression filter uses the structure of the neural network to propagate nonlinearities. Our experiments confirm that this leads to a stable algorithm. Unfortunately, it is difficult to prove theoretical convergence or stability bounds.

In the future, the cost function should be modified to give better results on long-term prediction. In addition our approach for the E-step should be compared to other filtering methods.

References

[1] 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.

[2] 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.

[3] S. Haykin, editor. *Kalman Filtering and Neural Networks.* Wiley Series on Adaptive and Learning Systems for Signal Processing. John Wiley & Sons, Inc., 2001.

[4] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, 2nd edition, 1998.

[5] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, pages 281–294, 1989.

[6] M. Cottrell, B. Girard, and P. Rousset. Forecasting of curves using classification. *Journal of Forecasting*, 17(5-6):429–439, 1998.

[7] 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.

[8] A. Sorjamaa, A. Lendasse, and M. Verleysen. Pruned lazy learning models for time series prediction. In *ESANN 2005, European Symposium on Artifical Neural Networks, Bruges (Belgium)*, pages 509–514, 2005.