European Symposium on
Time Series Prediction

7-8-9 February 2007
Espoo, Finland

Proceedings

European Symposium on
Time Series Prediction

ESTSP'07

7-8-9 February 2007
Espoo, Finland

Proceedings

Acknowledgements

The First European Symposium on Time Series Prediction is organized in collaboration with TKK (Helsinki University of Technology, Espoo), with financial support from:

- PASCAL network of excellence
- International Neural Networks Society
- European Neural Networks Society

Technical co-sponsors:

- International Neural Networks Society
- IEEE Computational Intelligence Society
- IEEE Region 8
- Pattern Recognition Society of Finland

The steering committee of ESTSP'07 would like to thank them for their appreciated contribution to the success of the conference.

Published by:
Multiprint Oy / Otamedia
Miestentie 3 a, FI-02150 Espoo, Finland
Phone: (09) 4393 200
Fax: (09) 4393 2020

Editor:
Amaury Lendasse

ISBN 978-951-22-8601-0

Foreword

Time series forecasting is a challenge in many fields. In finance, experts forecast stock exchange courses or stock market indices; data processing specialists forecast the flow of information on their networks; producers of electricity forecast the load of the following day.

The common point to their problems is the following: how can one analyze and use the past to predict the future?

The European Symposium on Time Series Prediction (ESTSP'07) is a new event in the fields of neural networks, statistics and econometrics. It is held on 7-9 February 2007 in Espoo (Helsinki), one of the most innovative towns in Europe. ESTSP'07 is a unique opportunity for researcher from statistics, neural networks, machine learning, control and econometrics to share their knowledge in the field of Time Series Prediction.

Forty-six papers have been submitted to ESTSP'07 and reviewed. The best thirty-two papers have been accepted. All the presentation will be oral. The selection was difficult due to the high scientific quality of the submitted papers.

We would like to thank all members of the scientific committee of ESTSP07 for their appreciated work in the reviewing process; they were helped by many colleagues, most of them remaining anonymous, and we associate them in our grateful thanks.

For the steering and local committee,
Amaury Lendasse

Steering and local committee

Francesco Corona	Helsinki Univ. of Technology (FI)
Marie Cottrell	Univ. Paris I (F)
Amaury Lendasse	Helsinki Univ. of Technology (FI)
Elia Liitiainen	Helsinki Univ. of Technology (FI)
Yoan Miche	Helsinki Univ. of Technology (FI)
Antti Sorjamaa	Helsinki Univ. of Technology (FI)
Michel Verleysen	UCL Louvain-la-Neuve (B)

Scientific committee

Eric de Bodt	Univ. Lille II (F) & UCL Louvain-la-Neuve (B)
Manfred Deistler	Vienna University of Technology (AUT)
Mark J. Embrechts	Rensselaer Polytechnic Institute (USA)
Manuel Grana	UPV San Sebastian (E)
Tom Heskes	Univ. Nijmegen (NL)
Jaakko Hollmen	Helsinki Univ. of Technology(FIN)
Christian Jutten	INPG Grenoble (F)
Juha Karhunen	Helsinki Univ. of Technology(FIN)
Samuel Kaski	Helsinki Univ. of Technology(FIN)
Erkki Oja	Helsinki Univ. of Technology (FIN)
Alberto Prieto	Universitad de Granada (E)
Jose Principe	University of Florida (USA)
Fabrice Rossi	INRIA (F)
Joseph Rynkiewicz	Univ. Paris I (F)
Francisco Sandoval	Univ.Malaga (E)
Eric Severin	Universite de Lille1 (F)
Olli Simula	Helsinki Univ. of Technology (FIN)
Jochen Steil	Univ. Bielefeld (D)
Johan Suykens	KUL Leuven (B)
Marc Van Hulle	KUL Leuven (B)
Thomas Villmann	Univ. Leipzig (D)
Vincent Wertz	UCL Louvain-la-Neuve (B)
Donald Wunsch	University of Missouri (USA)

ESTSP'07 Program

Wednesday, February 7, 2007

08:30 *Registration*

09:00 *Welcome* Amaury Lendasse and Olli Simula

Invited Talk
09:15 Manfred Deistler

10:15 *Coffee Break*

Session 1
10:45 S. Sarkka, A. Vehtari and J. Lampinen
11:10 M. Tornio, A. Honkela and J. Karhunen
11:35 G. Rubio, L.J. Herrera and H. Pomares
12:00 H. Sasaki, Y. Nakada, T. Kaburagi and T. Matsumoto

12:25 *Lunch Break*

Session 2
13:30 M. Nariman, M.B Menhaj and A. Mehrdad
13:55 J. Ibanez and G. Tunncliffe-Wilson
14:20 S. Rahat Abbas and M. Arif
14:45 F. Corona and A. Lendasse

15:10 *Coffee Break*

Session 3
15:40 T. Karna and A. Lendasse
16:05 J. Tikka and J. Hollmen
16:30 H. Peyvandi, A. Dalaki and C. Lucas

Thursday, February 8, 2007

Invited Talk

09:00	Michel Verleysen

10:00 *Coffee Break*

Session 4

10:20	M. Mirmomeni, C. Lucas and E. Ahmadi
10:45	Mirmomeni, B. Nadjar Araabi and C. Lucas
11:10	S. Kurogi, S. Tanaka and R. Koyama
11:35	M. Mirmomeni, M. Shafiee and C. Lucas

12:00 *Lunch Break*

Session 5

13:00	M. Olteanu and J. Rynkiewicz
13:25	L. J. Herrera, H. Pomares, I. Rojas, A.Guillen and G. Rubio
13:50	A. Sorjamaa and A. Lendasse

14:15 *Coffee Break*

Session 6

14:35	B. Perez-Sanchez, B. Guijarro-Berdinas and O. Fontenla-Romero
15:00	M.P. Cuellar, M. Delgado and M.C. Pegalajar
15:25	S. F. Crone and N. Kourentzes

16:00 *Social Events and Conference Dinner*

Friday, February 9, 2007

Invited Talk

09:00	Donald Wunsch
10:00	*Coffee Break*

Session 7

10:30	M.-G. M. Zulpukarov, G.G. Malinetskii and A.V. Podlazov
10:55	D.V Serebryakov, I.V. Kuznetsov and M.V. Rodkin
11:20	J. Arroyo, A. Munoz San Roque, C. Mate and A. Sarabia
11:45	H. Njimi and G. Melard
12:10	*Lunch Break*

Session 8

13:15	R. M. Kunst
13:40	B. Menezes, A. Seth and R. Singh
14:05	L. J. Herrera, H. Pomares, I. Rojas, G. Rubio and A.Guillen
14:30	E. Atukeren
14:55	*Coffee Break*

CONTENTS

Prediction of ESTSP Competition Time Series by Unscented Kalman Filter and RTS Smoother

Simo Särkkä, Aki Vehtari and Jouko Lampinen

Helsinki University of Technology
Department of Electrical and Communications Engineering
Laboratory of Computational Engineering
Box 9203, FIN-02015 HUT, Finland

Abstract. This article presents a solution to the time series prediction competition of the ESTSP 2007 conference. The solution is based on *optimal filtering*, which is a methodology for computing recursive solutions to statistical inverse problems, where a time varying stochastic state space model is measured through a sequence of noisy measurements. In the solution, the overall behavior of the time series is first modeled by constructing a linear state space model, which captures most of the visible features of the time series. Residual analysis techniques are then used for correcting the yet unmodeled features of the time series. These corrections result in a non-linear state space model, which is solved using a combination of linear Kalman filter, non-linear unscented Kalman filter and Rauch-Tung-Striebel smoother. The unknown parameters of the state space model are optimized to give the best possible prediction over 50 steps.

1 Introduction

In this article, a solution to the ESTSP 2007 time series prediction competition is presented. The solution is based on modeling the time series with a non-linear state space model, which is then estimated with unscented Kalman filter and Rauch-Tung-Striebel smoother. The solution is similar to the time series prediction method used in [1], but the underlying linear stochastic model is different and an additional non-linear correction term is included in the model.

1.1 Optimal Filtering and Smoothing

The celebrated *Kalman filter* [2, 3, 4] considers *optimal filtering*, that is, *recursive statistical inference* of linear state space models of the form

$$\begin{aligned} \mathbf{x}_k &= \mathbf{A}_{k-1}\,\mathbf{x}_{k-1} + \mathbf{q}_{k-1} \\ \mathbf{y}_k &= \mathbf{H}_k\,\mathbf{x}_k + \mathbf{r}_k, \end{aligned} \tag{1}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state, $\mathbf{y}_k \in \mathbb{R}^m$ is the measurement, $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q}_{k-1})$ is the process noise, and $\mathbf{r}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{R}_k)$ is the measurement noise. The matrix $\mathbf{A}_{k-1}$ is the transition matrix of the dynamic model and $\mathbf{H}_k$ is the measurement model matrix. In addition to producing the optimal estimates, the Kalman filter can be used for computing optimal n-step prediction of the state space model given a sequence measurements from the model.

The *Extended Kalman filter (EKF)* [3, 5, 6] and *Unscented Kalman filter (UKF)* [7, 8, 5] are extensions of the Kalman filter to estimation and prediction of non-linear state space models of the form

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, k-1) + \mathbf{q}_{k-1} \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, k) + \mathbf{r}_k, \end{aligned} \tag{2}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state, $\mathbf{y}_k \in \mathbb{R}^m$ is the measurement, $\mathbf{q}_{k-1} \sim \mathrm{N}(0, \mathbf{Q}_{k-1})$ is the Gaussian process noise, $\mathbf{r}_k \sim \mathrm{N}(0, \mathbf{R}_k)$ is the Gaussian measurement noise, $\mathbf{f}(\cdot)$ is the dynamic model function and $\mathbf{h}(\cdot)$ is the measurement model function.

Optimal smoothing methods [9, 10, 11] can be used for computing better estimates of the signal than optimal filters in the cases that the whole history of measurements from the time series can be used for computing the estimates. As optimal filtering methods produce optimal estimates, which are optimal when only causal estimators are considered, *optimal smoothing methods* produce optimal estimates based on the whole history of observations. Obviously, these estimates can be, in principle, computed from the posterior distribution of the states given the measurements, but the idea of optimal smoothing methods is to provide computationally efficient methods for computing these estimates.

1.2 Stochastic Differential Equations

Stochastic differential equation [12, 13] is a white noise driven differential equation of, for example, the form

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t) + \mathbf{L}(t)\,\mathbf{w}(t), \tag{3}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the (continuous-time) state, $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}_+ \mapsto \mathbb{R}^n$ is the drift function, $\mathbf{L}(t) \in \mathbb{R}^{n \times s}$ is the dispersion matrix, and $\mathbf{w}(t) \in \mathbb{R}^s$ is a white noise process with spectral density matrix $\mathbf{Q}_c(t)$.

The theory of stochastic differential equations is well known, and it is commonly formulated in terms of *Itô calculus*, which is the theory of differential calculus of stochastic processes (see, e.g., [12, 13]). In rigorous mathematical sense the stochastic differential equation (3) should be actually interpreted as a *stochastic integral equation* of the form

$$\mathbf{x}(t) - \mathbf{x}(s) = \int_s^t \mathbf{f}(\mathbf{x}, t)\,\mathrm{d}t + \int_s^t \mathbf{L}(t)\,\mathrm{d}\boldsymbol{\beta}(t), \tag{4}$$

which can be written more compactly as

$$\mathrm{d}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}, t)\,\mathrm{d}t + \mathbf{L}(t)\,\mathrm{d}\boldsymbol{\beta}(t), \tag{5}$$

where $\boldsymbol{\beta}(t)$ is a Brownian motion with diffusion matrix $\mathbf{Q}_c(t)$. If we define the *white noise* $\mathbf{w}(t)$ as the formal derivative of Brownian motion $\mathbf{w}(t) = \mathrm{d}\boldsymbol{\beta}(t)/\mathrm{d}t$, the equation (5) can be formally written in form (3). This kind of white noise

formulation only makes sense in the case when the dispersion matrix is independent of the state, that is, $\mathbf{L}(\mathbf{x}, t) = \mathbf{L}(t)$, because it is the case when the Itô and Stratonovich interpretations of the SDE are equivalent.

In this article, the dynamic model will be a *linear time invariant* (LTI) stochastic differential equation of the form

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{F}\,\mathbf{x}(t) + \mathbf{L}\,\mathbf{w}(t), \tag{6}$$

where $\mathbf{x}(t)$ is the state, $\mathbf{F}$ and $\mathbf{L}$ are constant matrices, and $\mathbf{w}(t)$ is a white noise process with a constant spectral density matrix $\mathbf{Q}_c$. The theory of LTI equations is much less complicated than the general Itô calculus and the relevant results of it can be found in beginning Chapters many estimation theory oriented books (e.g., [4, 6]).

2 Time Series Prediction

In this section the models and methods for time series prediction, and the prediction results are described. The model is constructed as follows:

1. First a linear state space model is constructed, which captures the dominant features of the time series.
2. A non-linear correction term is added to the model, which models the unmodeled non-linearity in the periodic signal.
3. The remaining auto-regressive component in the residual is compensated by fitting an AR-model to the residual.

The models are estimated using Kalman filter, unscented Kalman filter and Rauch-Tung-Striebel smoother.

2.1 Linear State Space Model

By looking at the time series data, which is shown in Figure 1, it can be seen to be sum of two main components, *a bias component* and *a periodic component*:

- The bias component can be modeled as a *Brownian motion*, which is formally the integral of continuous-time white noise $w_b(t)$, and it can be formulated as a solution of the stochastic differential equation model:

$$\frac{\mathrm{d}x_b}{\mathrm{d}t} = w_b(t). \tag{7}$$

- The periodic component can be modeled as a resonator with a certain angular velocity ω. The variations from perfect sinusoidal can be modeled by including a random white noise forcing term $w_r(t)$ to the resonator equation. This results in the stochastic differential equation model:

$$\frac{\mathrm{d}^2x_r}{\mathrm{d}t^2} = -\omega^2\,x_r + w_r(t). \tag{8}$$

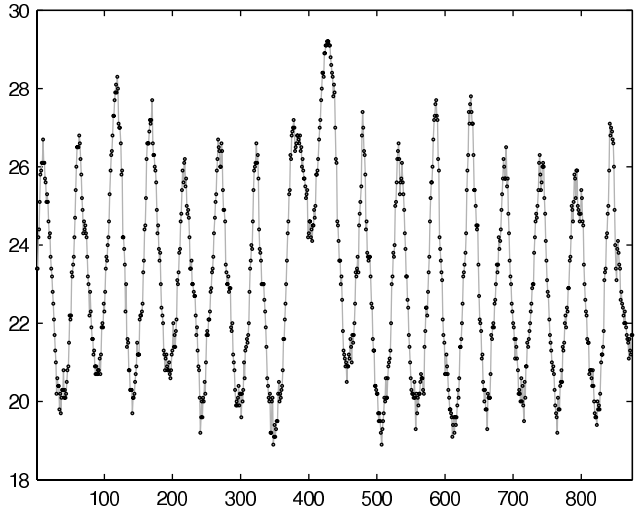

Fig. 1: ESTSP 2007 time series prediction competition data.

The actual time series is the sum of the bias and periodic components. In order to model the deviations of the data from the model a small Gaussian measurement noise r_k is assumed to be present in the measurements:

$$y_k = x_b(k) + x_r(k) + r_k, \tag{9}$$

where $k = 1, 2, \ldots$. The model can be equivalently written in form

$$\underbrace{\begin{pmatrix} \mathrm{d}x_b/\mathrm{d}t \\ \mathrm{d}x_r/\mathrm{d}t \\ \mathrm{d}^2x_r/\mathrm{d}t^2 \end{pmatrix}}_{\mathrm{d}\mathbf{x}/\mathrm{d}t} = \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -\omega^2 & 0 \end{pmatrix}}_{\mathbf{F}} \underbrace{\begin{pmatrix} x_b \\ x_r \\ \mathrm{d}x_r/\mathrm{d}t \end{pmatrix}}_{\mathbf{x}} + \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}}_{\mathbf{L}} \underbrace{\begin{pmatrix} w_b \\ w_r \end{pmatrix}}_{\mathbf{w}} \tag{10}$$

$$y_k = \underbrace{\begin{pmatrix} 1 & 1 & 0 \end{pmatrix}}_{\mathbf{H}} \underbrace{\begin{pmatrix} x_b \\ x_r \\ \mathrm{d}x_r/\mathrm{d}t \end{pmatrix}}_{\mathbf{x}} + r_k. \tag{11}$$

If we define the state as $\mathbf{x} = (x_b \; x_r \; \mathrm{d}x_r/\mathrm{d}t)^T$, the model can be seen to be a linear Gaussian continuous-discrete filtering model:

$$\begin{aligned} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} &= \mathbf{F}\,\mathbf{x}(t) + \mathbf{L}\,\mathbf{w}(t) \\ y_k &= \mathbf{H}\,\mathbf{x}(k) + r_k. \end{aligned} \tag{12}$$

By the well known formulas for linear systems (see, e.g., [4]) the transition matrix and covariance matrix of the equivalent discrete time model are given as

$$\begin{aligned}
\mathbf{A} &= \exp(\mathbf{F}) \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & \frac{\sin(\omega)}{\omega} \\ 0 & -\omega\,\sin(\omega) & \cos(\omega) \end{pmatrix} \\
\mathbf{Q} &= \int_0^1 \exp\Big((1-\tau)\,\mathbf{F}\Big)\,\mathbf{L}\,\mathbf{Q}_c\,\mathbf{L}^T\,\exp\Big((1-\tau)\,\mathbf{F}\Big)^T \mathrm{d}\tau \\
&= \begin{pmatrix} q_b & 0 & 0 \\ 0 & \frac{q_r\,\omega - q_r\,\cos(\omega)\,\sin(\omega)}{2\omega^3} & \frac{q_r\,\sin^2(\omega)}{2\omega^2} \\ 0 & \frac{q_r\,\sin^2(\omega)}{2\omega^2} & \frac{q_r\,\omega + q_r\,\cos(\omega)\,\sin(\omega)}{2\omega} \end{pmatrix},
\end{aligned} \tag{13}$$

where q_b and q_r are the spectral densities of w_b and w_r, respectively. If we denote the discrete-time state as $\mathbf{x}_k \triangleq \mathbf{x}(k)$, the model can be written as

$$\mathbf{x}_k = \mathbf{A}\,\mathbf{x}_{k-1} + \mathbf{q}_{k-1} \tag{14}$$

$$y_k = \mathbf{H}\,\mathbf{x}_k + r_k, \tag{15}$$

which is a linear state space model, and suitable for the Kalman filter.

The parameters ω, q_b and q_r can be estimated such that they give the best possible prediction as follows:

1. Find the places in the second half of the time series, which are similar to the place where the time series ends.
2. Select a discrete set of possible parameter values and predict 50 steps a head from the selected time series places.
3. Compute the errors in the predictions and select the parameter values that give the least total error when the predictions at all the selected time series places are summed.

The result of prediction with the estimated parameter values is shown in the Figure 2. The result is the smoothing result, computed by running Kalman filter and Rauch-Tung-Striebel smoother through the data. The estimated bias $x_b(t)$ is also separately shown in the figure. The measurement noise variance is set to an arbitrary value $\sigma_r^2 = 1$, because its value does not affect the prediction.

2.2 Non-Linear Correction Term

Although, the prediction of linear state space model in Figure 2 already captures the essential features of the time series, it is far from perfect. One feature, which can be seen in the time series is that the periodic component does not seem to be perfect sinusoidal, but possibly a non-linear transformation of a sinusoidal. This can be checked by plotting the measurements minus the estimated biases

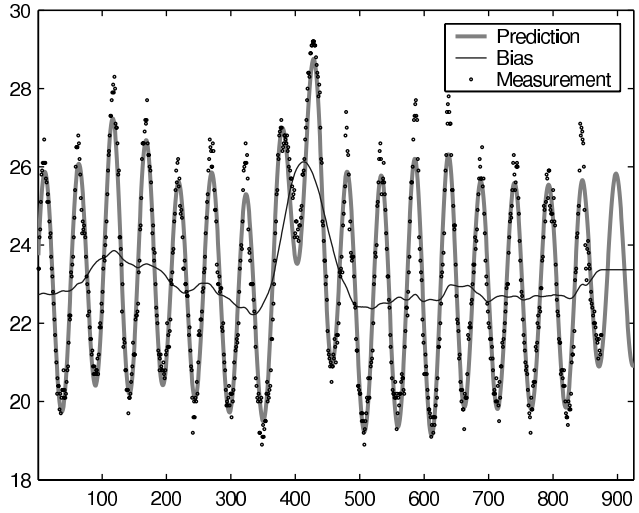

Fig. 2: Linear model smoothing and prediction result.

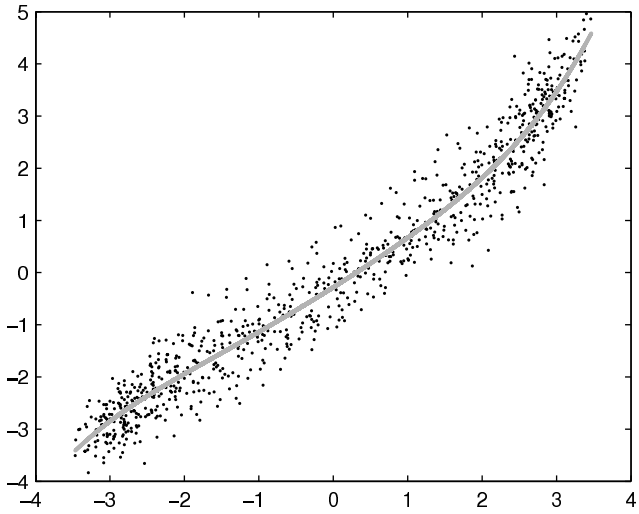

Fig. 3: Measurements minus the biases $y_k - \hat{x}_b$ as function of estimates of the resonator state $\hat{x}_r(k)$ and the fitted polynomial.

$y_k - \hat{x}_b$ as function of estimates of the resonator state $\hat{x}_r(k)$. The result is shown in the Figure 3 together with 5th order polynomial fitted to the function.

The fitted 5th order polynomial can be now included as part of the measure-

ment model as follows:

$$y_k = x_b(r) + \sum_{i=0}^{5} c_i \left(x_r(k) \right)^i + r_k, \tag{16}$$

where c_i are the coefficients of the polynomial. The measurement model is now non-linear, but fortunately of the form (2), which is suitable for the *unscented Kalman filter* (UKF).

In order to get an estimate of the time series based on all the measurements instead of the filtering estimate, which is based on the preceding data, Rauch-Tung-Striebel (RTS) smoother was ran over the UKF filtering result. Note that because the dynamic model is linear, linear RTS smoother is enough and there is no need to use a non-linear smoother.

2.3 Auto-regressive Model for Residual

The non-linear correction term in (16) models the signal-dependence of the residual quite well, but there still exists significant auto-correlation in the residual as can be seem in the Figure 4. There seems to be a small periodic component in the residual, which needs to be compensated.

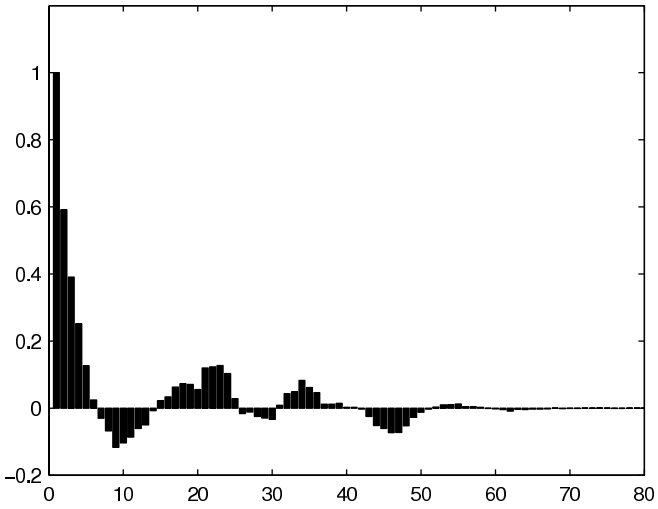

Fig. 4: Auto-correlation of the residual.

The periodicity of the residual time series $\{e_k : k = 1, ..., N\}$ can be modeled with a second order auto-regressive (AR) model [14]

$$e_k = \sum_{i=1}^{2} a_i \, e_{k-i} + r_k^{\mathtt{ar}}. \tag{17}$$

The order of the model was selected to be two, because a second order AR-model is able to capture the single periodic component in the residual and there is no evidence of higher order phenomena. In article [1], an AR-model of the same order was also successfully applied to modeling similar residual periodicity. The variance of the Gaussian noise $r_k^{\mathtt{ar}}$ was set to a suitable value $\sigma_{\mathtt{ar}}^2 = 1$. The coefficients of the AR-model were estimated with the linear least squares method.

After the AR-model has been estimated from the residual time series data, the final estimation solution is obtained by running the Kalman filter and the Rauch-Tung-Striebel smoother to the model

$$\begin{aligned} d_k &= \sum_{i=1}^{2} a_i \, d_{k-i} + v_k^{\mathtt{p}} \\ e_k &= d_k + r_k^p, \end{aligned} \tag{18}$$

where the Gaussian process noise $v_k^{\mathtt{p}}$ has variance $q^{\mathtt{p}} = 10^{-8}$ and the measurement noise r_k^p has the variance $\sigma_p^2 = 10^{-9}$. The residual estimate $\hat{d}_k$ is summed to the prediction computed by the UKF.

2.4 Final Prediction Result

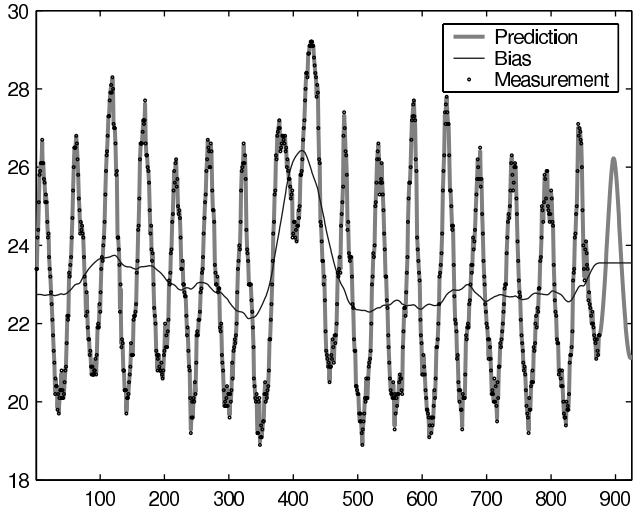

Fig. 5: The final prediction result.

The final prediction result is shown in the Figure 5. The prediction was obtained as follows:

1. A high number of parameter values were selected and the values were searched, which gave the best prediction according to the principle described in Section 2.1.

2. With each parameter values, the non-linear state space model described in Section 2.2 with the linear dynamic model (14) and non-linear measurement model (16) was estimated with *unscented Kalman filter (UKF)* and the prediction was done by iterating the prediction step of the filter 50 times. The non-linear correction in the prediction was applied only to the predicted mean, which corresponds to EKF type of approximation. This was selected, because according to the experiments it gave a lower prediction error than the UKF type of approximation. In the estimation step the UKF worked better than EKF.

3. The Rauch-Tung-Striebel (RTS) smoother was run over the time series to get an estimate, which is conditioned to all the measurements. The auto-regressive model was fitted to the smoother residual and the auto-regressive correction was then added to the prediction results as described in Section 2.3. This estimation was done using linear Kalman filter and RTS smoother.

2.5 Discussion

The approach used in this article is *exploratory* in the sense that a simple linear model is first constructed and it is then improved step by step. The approach is not incremental, because at the final step the whole non-linear model is estimated and used as whole instead of using the results of the models on the previous steps as such.

The Rauch-Tung-Striebel smoother is only used in estimation stage, and actually, it is only needed for computation of the residuals of the solutions. The actual prediction is performed by using the prediction step of the unscented Kalman filter and the non-linear correction is applied to the prediction result. However, the smoother result is needed for estimating the non-linear residual correction term and the AR-model of the residual.

The selection of parameters is based on *predictive criterion*, which tries to mimic the competition error criterion as well as possible. The idea of the approach is to find the parameter values, which minimize the expected value of the prediction error in the competition, when the expected value is computed over the posterior distribution of the other parameters in the model. This can be interpreted as optimal Bayesian decision [15] for selection of point estimates for the parameters. In this case it is not optimal to use, for example, maximum a posterior (MAP) estimate, minimum mean squared error (MMSE) estimate, that is, the posterior mean or any other such point estimates, because they minimize wrong error criteria. Instead, it is best to explicitly minimize the 50 step prediction error criterion of the competition.

3 Conclusion

In this article a solution to the time series prediction competition of the ESTSP 2007 conference has been presented. The solution is based on constructing a non-

linear state space model for the time series, which is then estimated using the unscented Kalman filter (UKF) and Rauch-Tung-Striebel (RTS) smoother. The residual auto-correlations were compensated with an AR-model. The parameters were selected by systematically testing various combinations of parameters and by selecting the ones, which gave the least error in 50 step prediction.

References

[1] Simo Särkkä, Aki Vehtari, and Jouko Lampinen. Time series prediction by Kalman smoother with cross-validated noise density. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1653–1658, July 2004.

[2] Rudolph E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, 82:35–45, March 1960.

[3] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[4] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering, Theory and Practice Using MATLAB*. Wiley Interscience, 2001.

[5] Simon Haykin, editor. *Kalman Filtering and Neural Networks*. Wiley, 2001.

[6] Yaakov Bar-Shalom, Xiao-Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley Interscience, 2001.

[7] Simon J. Julier, Jeffrey K. Uhlmann, and Hugh F. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proceedings of the 1995 American Control, Conference, Seattle, Washington*, pages 1628–1632, 1995.

[8] Simon J. Julier and Jeffrey K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, March 2004.

[9] H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, 1965.

[10] J.S. Meditch. *Stochastic optimal linear estimation and control*. McGraw-Hill New York, 1969.

[11] Andrew P. Sage and James L. Melsa. *Estimation Theory with Applications to Communications and Control*. McGraw-Hill Book Company, 1971.

[12] Ioannis Karatzas and Steven E. Shreve. *Brownian Motion and Stochastic Calculus*. Springer, 1991.

[13] Bernt Øksendal. *Stochastic Differential Equations: An Introduction with Applications*. Springer, 6 edition, 2003.

[14] Monson H. Hayes. *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, Inc., 1996.

[15] José M. Bernardo and Adrian F. M. Smith. *Bayesian Theory*. John Wiley & Sons, 1994.

Time Series Prediction with Variational Bayesian Nonlinear State-Space Models

Matti Tornio, Antti Honkela, and Juha Karhunen

Adaptive Informatics Research Centre, Helsinki University of Technology
P.O. Box 5400, FI-02015 TKK, Finland
{Matti.Tornio, Antti.Honkela, Juha.Karhunen}@tkk.fi
http://www.cis.hut.fi/projects/bayes/

Abstract. In this paper the variational Bayesian method for learning nonlinear state-space models introduced by Valpola and Karhunen in 2002 is applied to prediction in the ESTSP'07 time series prediction competition data set. The data set is pre-processed by approximately removing the periodic component of the data and the nonlinear state-space model is only learned on the residuals. The model uses multilayer perceptron (MLP) networks to model the nonlinearities of the system which allows the modelling of complex dynamical processes. The variational Bayesian learning approach is resistant to overfitting and allows comparison of different model structures using the derived lower bound on marginal log-likelihood. The desired predictions are evaluated as the mean of a Monte Carlo approximation of the predictive distribution.

1 Introduction

Traditionally, time series prediction is done using models based directly on the past observations of the time series. Perhaps the two most important classes of neural network based solutions used for nonlinear prediction are feedforward autoregressive neural networks and recurrent autoregressive moving average neural networks [10]. However, instead of modelling the system based on past observations, it is also possible to model the same information in a more compact form with a state-space model [3].

This paper uses the nonlinear state-space model (NSSM) introduced by Valpola and Karhunen in 2002 [11] to model a time series. The primary goal of the paper is to apply this publicly available[1] NSSM to the task of time-series prediction as a black box tool.

The nonlinearities of both the dynamics and the mapping from the states to observations are modelled with multilayer perceptron (MLP) networks. Training a nonlinear state-space model is a computationally challenging task and prone to overfitting. The NSSM in [11] uses variational Bayesian learning, which is both resistant against overfitting and computationally effective compared to e.g. sampling methods.

[1] http://www.cis.hut.fi/projects/bayes/software/

2 Nonlinear State-Space Models by Variational Bayesian Learning

2.1 The Model

The variational Bayesian nonlinear state-space model introduced by Valpola and Karhunen in [11] uses a general nonlinear state-space model for the observations $\mathbf{x}(t)$

$$\mathbf{s}(t) = \mathbf{g}(\mathbf{s}(t-1), \boldsymbol{\theta}_\mathbf{g}) + \mathbf{m}(t) \tag{1}$$
$$\mathbf{x}(t) = \mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) + \mathbf{n}(t) \tag{2}$$

with states $\mathbf{s}(t)$, Gaussian innovation $\mathbf{m}$ and noise $\mathbf{n}$, and multi-layer perceptron (MLP) networks to model the nonlinearities $\mathbf{f}$ and $\mathbf{g}$. The functional form of the MLP networks is given by

$$\mathbf{g}(\mathbf{s}(t-1), \boldsymbol{\theta}_\mathbf{g}) = \mathbf{s}(t-1) + \mathbf{D}\tanh(\mathbf{C}\mathbf{s}(t-1) + \mathbf{c}) + \mathbf{d} \tag{3}$$
$$\mathbf{f}(\mathbf{s}(t), \boldsymbol{\theta}_\mathbf{f}) = \mathbf{B}\tanh(\mathbf{A}\mathbf{s}(t) + \mathbf{a}) + \mathbf{b}, \tag{4}$$

where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ are the network weight matrices and $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$ are the bias vectors. Inference and learning in the model can be made more reliable and efficient than in [11] by using the new linearisation described in [5].

2.2 Variational Bayes

Variational Bayesian learning [7, 2] is based on approximating the posterior distribution $p(\boldsymbol{\theta}, \boldsymbol{S}|\boldsymbol{X}, \mathcal{H})$ with a tractable approximation $q(\boldsymbol{\theta}, \boldsymbol{S}|\boldsymbol{\xi})$, where $\boldsymbol{X} = \{\mathbf{x}(t)|t = 1, \ldots, T\}$ is the data, $\boldsymbol{S} = \{\mathbf{s}(t)|t = 1, \ldots, T\}$ are the latent state values, $\boldsymbol{\theta}$ are the parameters of the model $\mathcal{H}$, and $\boldsymbol{\xi}$ are the (variational) parameters of the approximation. The approximation is fitted by maximising a lower bound on marginal log-likelihood

$$\mathcal{B} = \left\langle \log \frac{p(\boldsymbol{X}, \boldsymbol{S}, \boldsymbol{\theta}|\mathcal{H})}{q(\boldsymbol{S}, \boldsymbol{\theta}|\boldsymbol{\xi})} \right\rangle = \log p(\boldsymbol{X}|\mathcal{H}) - D_{\mathrm{KL}}(q(\boldsymbol{S}, \boldsymbol{\theta}|\boldsymbol{\xi})||p(\boldsymbol{S}, \boldsymbol{\theta}|\boldsymbol{X}, \mathcal{H})), \tag{5}$$

where $\langle\cdot\rangle$ denotes expectation over q. This is equivalent to minimising the Kullback–Leibler divergence $D_{\mathrm{KL}}(q||p)$ between q and p [6, 2].

The posterior approximations for the network weights and biases, as well as all the other model parameters except latent states are modelled as Gaussian distributions with a diagonal covariance. The posterior approximation for the latent states is modelled as a Gaussian distribution with an almost diagonal covariance. The correlation between the corresponding components $s_j(t)$ and $s_j(t-1)$ of subsequent state vectors is modelled, however. This is a realistic minimal assumption for modelling the dynamical system and does not increase the computational cost significantly [11].

2.3 Learning

The nonlinear state-space model is learned by numerically maximising the bound (5). This optimisation requires evaluating the value of the bound and its gradient with respect to all the variational parameters $\boldsymbol{\xi}$. To speed up this optimisation, a conjugate

gradient method is used to update the variational parameters of the latent states and the MLP network weights and biases instead of the heuristic algorithm presented in [11]. The other model parameters are updated as described in [11].

At the beginning of the learning, the network weights and biases are initialised to random values drawn from a Gaussian distribution. The latent states are initialised to the first principal components of embedded data vectors [11]. To ensure that the learning does not get stuck in a local minimum early on, the latent states and the model hyperparameters are not updated until the network weights and biases have converged to reasonable values. It is also useful to use multiple different initialisations to avoid local minima.

Modelling noise as part of the state-space model means that the model can filter out most of the noise in the original data set. Dynamics of these smoothed-out observations are often easier to learn than the dynamics of the original data set. State-space based approach can also typically model the system in a more compact form than a neural network model based directly on the past observations.

The variational Bayesian approach also provides a straightforward way to perform model selection. The lower bound on marginal log-likelihood $\mathcal{B}$ can be used as a measure of model quality between models with different structure such as different number of hidden units or different dimensionality of the state-space. Even if there is not enough data for methods such as cross-validation, this lower bound can still be used to evaluate relative model quality [11].

3 Time series prediction

Given data $\boldsymbol{X}$ and background assumptions $\mathcal{H}$, the optimal way to make predictions of an unknown quantity y with respect to mean-squared error is to use the mean of the posterior predictive distribution $p(y|\boldsymbol{X}, \mathcal{H})$ as the point prediction [1].

The easiest way to compute predictions of future observations based on the NSSM is simply to iterate Equation (1) starting from the posterior mean of the latent states corresponding to the last observed data sample. In some cases it can be desirable to ignore the innovation process $\mathbf{m}(t)$ (process noise) while doing these computations, as long predictions can lead to very high variance and the mean values of the predictions thus converge to the long term mean over very long prediction windows.

Even though the same techniques that are used in learning can also be used to compute the predictions, sampling methods typically lead to more accurate inference. Using the same approximation to evaluate $\mathbf{g}(\mathbf{s}(t-1), \boldsymbol{\theta}_\mathbf{g})$ as in learning consecutively leads to severe underestimation of predictive variance because the parameters $\boldsymbol{\theta}_\mathbf{g}$ used in consecutive steps would be assumed to be two independent sets even though they are the same. This is not a problem in learning which only requires one-step prediction, but for accurate long-term prediction the sampling approach is necessary.

For this purpose, the state values corresponding to the last observed data sample as well as all the network weights are sampled repeatedly from the variational posterior approximations, and the relevant predictions are evaluated using Eqs. (1) and (2) iteratively. This can be computationally much more demanding than using the same

procedure as in learning, but in most cases the time required for sampling is still insignificant compared to the training time of the original model.

In more complicated situations than direct prediction of future values, more advanced inference methods are needed to take into account the available future observations. This inference can be made more efficient by the method described in [9]. An example of this approach is given in [8], where the NSSM described in this paper is used to make predictions for a cart-pole system and at each time instant the new latent states are inferred from the latest observations and the future is predicted based on the model and the control signal.

4 Experiments: Prediction competition

The data set in the experiment was the prediction competition data set for ESTSP'07 [2]. This is a one-dimensional time series with 875 samples. The data set appears strongly periodic with a period of approximately 52 samples. To make prediction of the time series easier, the data set is averaged over all the full periods (samples from 1 to 832) and this average is subtracted from the original data set.

After this preprocessing, a state-space with three dimensions was used to model the dynamics of the residual time series. A three-dimensional state-space was chosen because it resulted in the best value for the bound on marginal log-likelihood $\mathcal{B}$. Both the observation MLP network and the dynamical MLP network had 20 hidden units. During 400 first iterations of the learning, an embedded version of the data set was used as described in [11]. The embedded data vector was $\hat{\mathbf{x}}(t) = [\mathbf{x}^T(t)\ \mathbf{x}^T(t-1)\ \mathbf{x}^T(t-2)\ \mathbf{x}^T(t-4)\ \mathbf{x}^T(t-8)\ \mathbf{x}^T(t-16)]^T$. The latent states were initialised to the three first principal components of the embedded data vector. The learning of the model took about three hours on a 2.2 GHz AMD Opteron processor.

A short overview of the preprocessing and prediction algorithm for a periodic time series $\mathbf{x}$ with length T, an approximated period T_{per} and a number of full periods N_{per} can be seen in Table 1.

The predictions made using the sampling method with 1000 particles for the next 61 time steps can be seen in Fig. 1. The predictions were computed with the innovation process ignored. The prediction length of 61 time steps was chosen so that the data set with the predictions contains 18 full periods of 52 samples. The reconstruction of the residual data set based on the model can be seen in Fig. 2. The latent state-space can be seen in Fig. 3. As some of the state components appear clearly periodic, it is likely that the period of 52 samples used in preprocessing was slightly incorrect. The original data set may also have contained components with longer periods.

From the Figs. 2 and 3 it is clear that the model of the dynamics of the residual system has a large associated uncertainty. This is natural, as the residual data set seen in Fig. 2 is quite hard to predict as it appears to have very little structure and there is little data compared to the very broad prior over different models. This uncertainty can also be seen in the predictions of the states that are very close to the long-term mean, along with large error bars for the first two states. These large error bars do not affect

[2] Available at http://estsp2007.org/files/competition_data.txt

Table 1: Prediction algorithm for periodic data.

Learning:
1. Compute the periodic component $\mathbf{x}_{per}$ over the full periods. The periodic component is given by $\mathbf{x}_{per}(i) = \frac{1}{N_{per}} \sum_{j=1}^{N_{per}} \mathbf{x}(\mathrm{mod}(i, T_{per}) + (j-1) \cdot T_{per})$, where we define $\mathrm{mod}(a \cdot n, n) = n$
2. Subtract the periodic component from the original data set $\mathbf{x}$. The resulting residual data set for each $i = 1 \ldots T$ is given by $\mathbf{x}_{res}(i) = \mathbf{x}(i) - \mathbf{x}_{per}(i)$
3. Use the NSSM to learn a state-space representation for the residual data set $\mathbf{x}_{res}$
Prediction:
4. Sample the initial state for the prediction $\mathbf{s}(T)$ and the network parameters $\boldsymbol{\theta}_g$ and $\boldsymbol{\theta}_f$ from the model learned in step 3
5. Iterate Equations (1) and (2) using the values sampled at step 4
6. Add the periodic component back to the predicted samples to get the final predictions

the predictions of the output, as the contribution of the third state to it is roughly 1000 times larger than those of the first two.

5 Discussion

The NSSM in [11] has been previously applied to several difficult prediction problems. One such example is the prediction of the dynamics of a complex system consisting of two Lorenz processes and a harmonic oscillator described in [11]. In [8], the model was used to predict the dynamics of a cart-pole system and the predictions were then used by a nonlinear model predictive controller. Even though the NSSM is better suited to modelling higher dimensional systems, it can also be used for modelling one-dimensional time series as in this paper.

The state-space model from [11] requires that the data set is evenly sampled. However, the recent extension of the model to continuous-time described in [4] allows the prediction of unevenly sampled time series as well. Continuous-time models also allow modelling both the short-term and long-term dynamics of the system more easily.

In theory the NSSM could have been used to predict the original data set without any preprocessing. However, with the limited amount of available data and a flexible prior over a large space of possible nonlinear models, there would have been significant posterior uncertainty on the dynamics and the global prediction would soon have converged to the long-term mean with large variance. In order to attain more meaningful predictions, more prior information such as the apparent periodicity of the signal have to be taken into consideration.

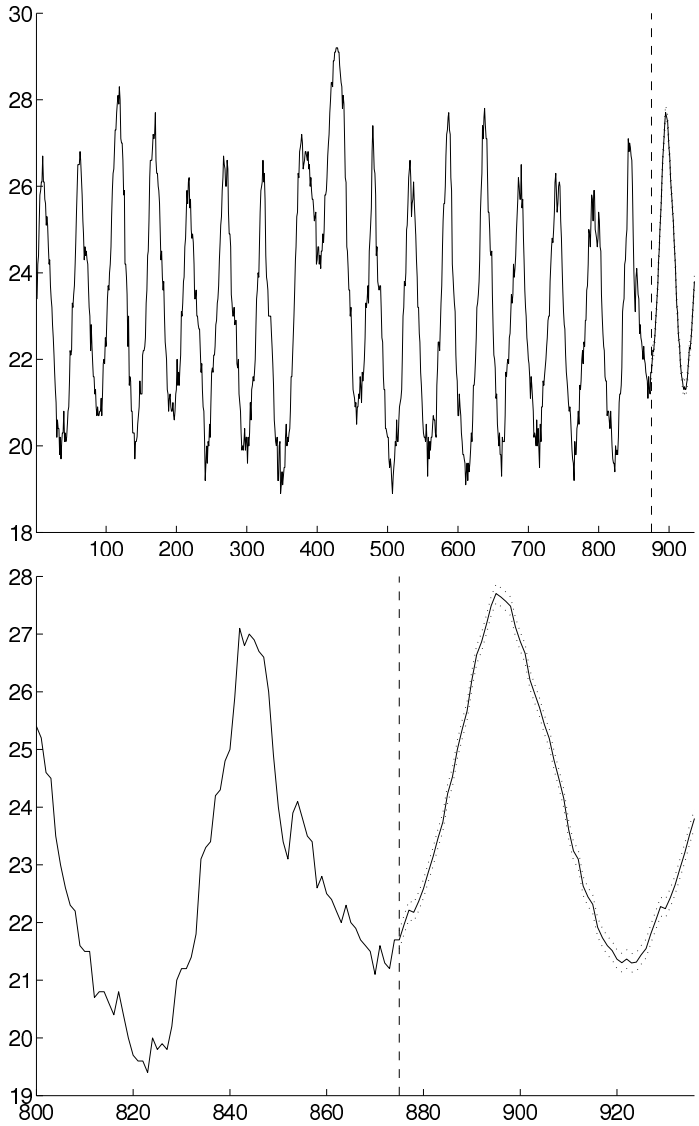

Fig. 1: Top: The original time series and the predicted 61 next time steps. Bottom: The original time series starting from time instant 800 and the predicted 61 next time steps. The dotted lines in both figures represent pseudo 95 % confidence intervals. Note that the intervals are smaller than in reality as the variance caused by the innovation is ignored.

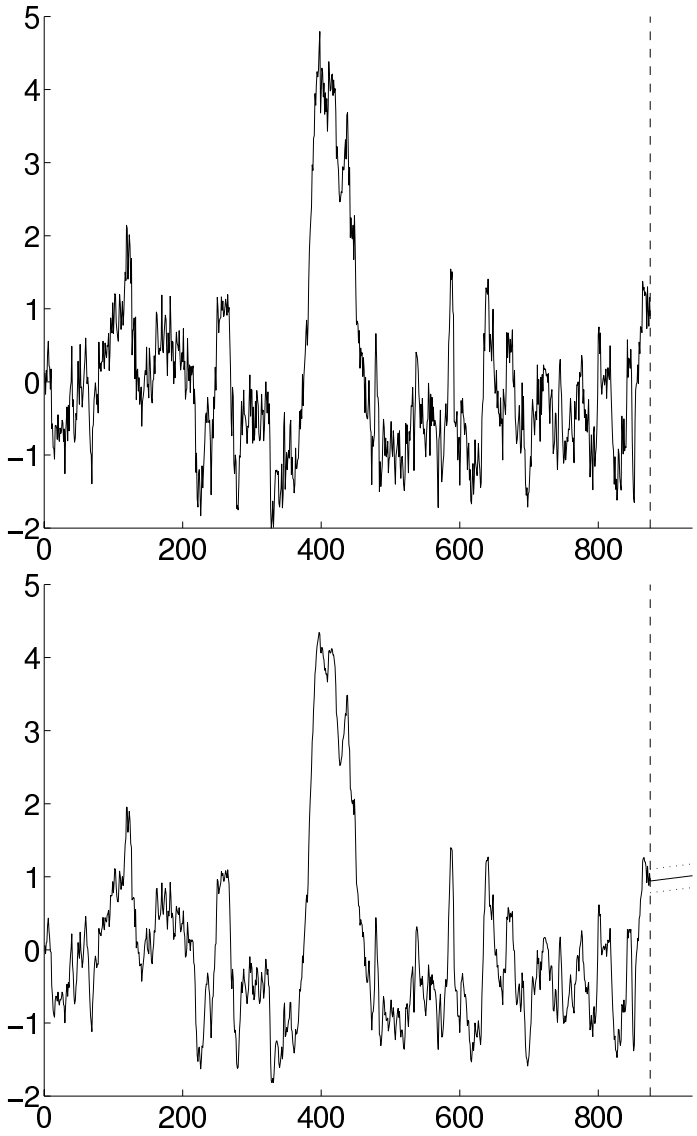

Fig. 2: Top: The original residual data set. Bottom: The mean of the reconstruction of the residual data set based on the model and its predictions. The reconstructed data set is the original data set with the observation noise filtered out. The dotted lines represent pseudo 95 % confidence intervals. The intervals are again smaller than in reality as the variance caused by the innovation is ignored.

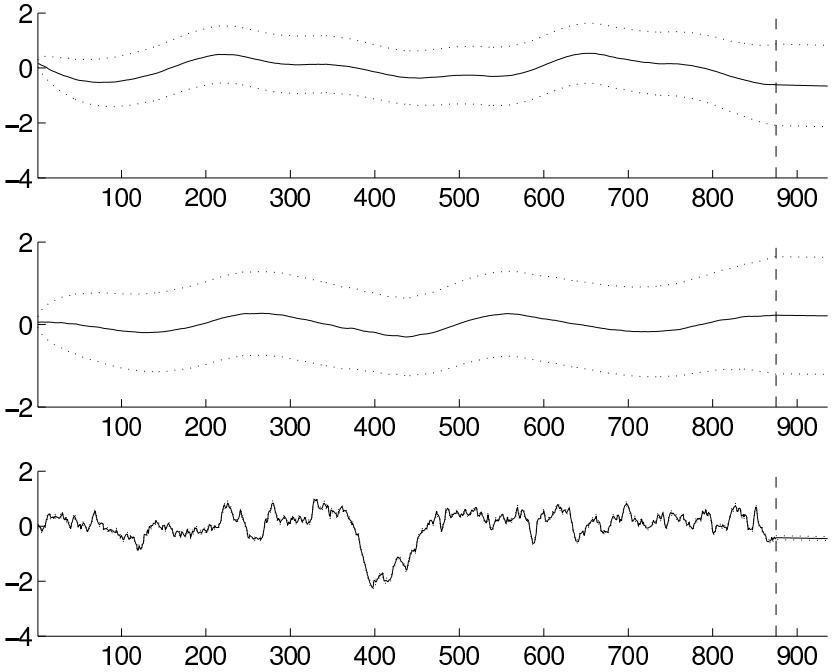

Fig. 3: The three dimensional latent state-space. Each of the three components of the state vector and their predictions are shown in its own figure. The dotted lines represent pseudo 95 % confidence intervals. The intervals are again smaller than in reality as the variance caused by the innovation is ignored.

6 Conclusion

In this paper we have applied the variational Bayesian NSSM of Valpola and Karhunen [11] to time series prediction. The prediction results with the ESTSP'07 prediction competition data set are presented.

Using state-space models for time series prediction has several benefits. The use of latent states allows easy handling of noisy data as the noise can be filtered out of the latent states. The state-space also allows creating models for partially observed systems, where some of the observations are not available. Finally, state-space models can usually represent the dynamics of the model in a more compact form than a model based directly on the past observations. Using variational Bayesian methods for learning these NSSMs is both resistant against overfitting and provides a cost function which can be used for model comparison.

Acknowledgments

The authors would like to thank Tapani Raiko for fruitful discussions. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

[1] J. M. Bernardo and A. F. M. Smith. *Bayesian Theory*. J. Wiley, 2000.

[2] C. Bishop. *Pattern Recognition and Machince Learning*. Springer, Cambridge, 2006.

[3] S. Haykin. *Neural Networks – A Comprehensive Foundation, 2nd ed.* Prentice-Hall, 1999.

[4] A. Honkela, M. Tornio, and T. Raiko. Variational Bayes for continuous-time nonlinear state-space models. In *NIPS*2006 Workshop on Dynamical Systems, Stochastic Processes and Bayesian Inference*, Whistler, B.C., Canada, 2006.

[5] A. Honkela and H. Valpola. Unsupervised variational Bayesian learning of nonlinear models. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 593–600. MIT Press, Cambridge, MA, USA, 2005.

[6] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. In M. Jordan, editor, *Learning in Graphical Models*, pages 105–161. The MIT Press, Cambridge, MA, USA, 1999.

[7] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[8] T. Raiko and M. Tornio. Learning nonlinear state-space models for control. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN'05)*, pages 815–820, Montreal, Canada, 2005.

[9] T. Raiko, M. Tornio, A. Honkela, and J. Karhunen. State inference in variational Bayesian nonlinear state-space models. In *Proceedings of the 6th International Conference on Independent Component Analysis and Blind Source Separation (ICA 2006)*, pages 222–229, Charleston, South Carolina, USA, March 2006.

[10] A. Trapletti. *On Neural Networks as Statistical Time Series Models*. PhD thesis, Technische Universität Wien, 2000.

[11] H. Valpola and J. Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural Computation*, 14(11):2647–2692, 2002.

Gaussian Process regression applied to time series prediction: a particular case study in ESTSP 2007 time series prediction competition

G. Rubio, L.J. Herrera, H. Pomares

Dept. of Architecture and Technology Computer University of Granada - Spain

Abstract. Gaussian Process regression is a technique with good theoretical properties, that has been suggested as a replacement for supervised neural networks in non-linear regression. Although Gaussian Process regression is not widely used, its application to time series forecasting would be straightforward . A study of several Gaussian Process models is carried out over the data set of time series prediction competition in ESTSP 2007.

1 Introduction

Since Neal, in 1996 [2], showed that a RBF (Radial Basis Function) neural network converges to a Gaussian Process model when number of units in its hidden layer tends to infinity, Gaussian Process (GP) regression acquired importance in the machine learning field. Moreover, he also showed that a large class of neural networks behave in the same way. A solid mathematical background and the ease of obtaining and expressing uncertainty in predictions are desirable properties for a number of applications.

GP models depend on a kernel function (covariance function). Hence GP regression may be classified as a kernel technique, like Support Vector Machines ([1]). GP regression has its roots in the Bayesian approach to neural networks modelling. It allows a noise model and a prior over functions to be specified in a practical way via covariance function parameters, called hyperparameters in Bayesian framework nomemclature. One advantage of GP formulation over neural networks is that the combination of the prior and noise models can be exactly carried out using matrix operations. Hyperparameters can be estimated from data, by optimizing GP with respect to a given criteria (traditionally, reaching maximum likelihood [2],[6]).

2 Gaussian Process regression

A stochastic process $\{x_t\}$ $t \in T$ is Gaussian if and only if for all finite index subset $t_1, ..., t_k$ from index set T, $X_{t_1,...,t_k} = (x_{t_1}, ..., x_{t_k})$ is a random variable with Gaussian distribution.

GPs are defined by a covariance $C(x, x')$ and mean $u(x)$ functions. Only GPs where $u(x) = 0$ will be considered. Chosen covariance functions must generate positive definite matrix for input data $X^{N+1} = \{x_n^N\}_{n=1}$.

Training data consists of n inputs pairs and targets $(x_i^N, t_i), i = 1...n$, where x_i is a d-dimensional vector. We denote by C_N the covariance matrix generated

by the covariance function C with training data $C_N(i,j) \equiv C(x_i^N, x_j^N)$ and by t_N the input targets vector $t_N = (t_1, ..., t_n)^T$. For new data to be predicted, result will be a Gaussian distribution with mean and variance:

$$\hat{t}_{N+1} = K^T C_N^{-1} t_N \tag{1}$$

$$\hat{\sigma}^2_{\hat{t}_{N+1}} = U - K^T C_N^{-1} K \tag{2}$$

where $K(i,j) = C(x_i^N, x_j^{N+1})$, $U(i,j) = C(x_i^{N+1}, x_j^{N+1})$, and $x_i^{N+1} \in X_{N+1}$, the new data to be predicted. Predictive mean $\hat{t}_{N+1}$ for test cases is interpreted as predicted values and standard deviation $\hat{\sigma}_{\hat{t}_{N+1}}$ defines the error bars on this prediction.

2.1 Training a Gaussian Process

Given a covariance function $C(x, y; \Theta)$, where Θ are the hyperparameters $\Theta = (\theta_1, \theta_2, ..., \theta_m)$, we want to learn these hyperparameters from the training data $D = (X_N, T_N)$. In maximum likelihood framework, we adjust the hyperparameters to maximize the log likelihood or, equivalently, minimize negative log likelihood of the hyperparameters:

$$NLE = -logP(D|\Theta) = \log\det C_N + T_N^T C_N^{-1} T_N \tag{3}$$

but as Chapelle [8] remarks if we not trust the prior assumptions (i.e. covariance function, prior over the hyperparameters values) cross-validation criteria can be used, although they lead to more difficult optimization problems:

- Negative log predictive leave-one-out (NLP-LOO): NLE version with cross-validation.

$$NLP - LOO = \sum \log(C_N)_{ii} + \frac{(C_N^{-1}T)_i^2}{(C_N^{-1})_{ii}^2} \tag{4}$$

- Mean squared error leave-one-out (MSE-LOO): ignores predictive variance.

$$MSE - LOO = \sum \frac{(C_N^{-1}T)_i^2}{(C_N^{-1})_{ii}^2} \tag{5}$$

Priors over functions are expressed by a covariance function, but the prior over the hyperparameters can also be incorporated in selection criteria. For instance, if we prefer that hyperparameter values remain as low as possible (very common condition in regularization), the following expression can be added to the criteria:

$$prior(\Theta) = \Theta\Theta^T \tag{6}$$

2.2 Covariance functions

Covariance functions express in some way beliefs about the function we are modelling. As already said, the only restriction over the covariance function is that it must generate a positive definite matrix for input data X_N. Commonly used covariance functions are the following:

$$C(x^{(n)}, x^{(n')}; \{\theta_1, \theta_2, r\}) = \theta_1 \exp\left[-r\|x^{(n')} - x^{(n)}\|^2\right] + \theta_2 \tag{7}$$

$$C(x^{(n)}, x^{(n')}; \{\sigma_w, \sigma_c\}) = \sigma_w \langle x, x' \rangle + \sigma_c \tag{8}$$

$$C(x^{(n)}, x^{(n')}; \{\theta_1, \theta_2, r_{i=1}^d\}) = \theta_1 \exp\left[-\frac{1}{2}\sum_{i=1}^{d} r_i (x_i^{(n')} - x_i^{(n)})^2\right] + \theta_2 \tag{9}$$

$$C(x^{(n)}, x^{(n')}; \{\theta_1, r_{i=1}^d, \sigma_w, \sigma_c\}) = \theta_1 \exp\left[-\frac{1}{2}\sum_{i=1}^{d} r_i (x_i^{(n')} - x_i^{(n)})^2\right] + \sigma_w \langle x, x' \rangle + \sigma_c \tag{10}$$

In practice function 10 has given good results [3], [6].

A noise model can be expressed by adding a term to the diagonal of the covariance matrix or, equivalently, a term added to the covariance function:

$$C_{nm} = C(x^{(n)}, x^{(m)}; \Theta) + \delta_{nm} Z(x^{(n)}; \Theta) \tag{11}$$

where Z is a noise model and $\delta_{nm} = 1$ if $n = m$, and $\delta_{nm} = 0$ if $n \neq m$. For instance, input independent additive Gaussian noise is expressed by $Z(x^{(n)}; \Theta) = \theta_n$, a new parameter that can be incorporated in the model to be optimized.

3 Experiments

The data set of the time series prediction competition in ESTSP 2007 is the only information source we have, being the generating process totally unknown. A simple strategy was adopted for the study: dataset was normalized setting its mean to 0 and its variance to 1 and common beliefs and preference about the function to be approximated and the model to find were assumed:

- there are additive Gaussian noise
- parameters with low values are preferable for the model (regularization)

We also decided to use covariance functions from 7 (Gaussian), 8 (Linear) and 10 (Gaussian-Linear) with selection criteria for adaptation Negative Log Evidence (NLE, see 3) and Mean Square Error Leave-One-Out (MSE-LOO, see 5) in order to compare them.

3.1 Experiments results

For experiments, time serie data set was split in 2, the first 750 and last 125. Input/output vectors were created with 15 regressors $(t-15, ..., t-2, t-1)$ and prediction horizon of $t+1$ for each new data set, obtaining 735 vectors for training and 110 for test. GP models were adapted by a gradient conjugate algorithm with restarting in case of reaching local minima for a fixed number of iterations (1000). Training MSEs (Mean Square Error) for the different model selection criterias are chown in tables 1 and 2.

Covariance function	training MSE	test MSE
Linear	0.1640	0.1907
Gaussian	0.0167	0.2374
Gaussian-linear	0.0697	0.1732

Table 1: Results with selection criteria NLE

Covariance function	training MSE	test MSE
Linear	0.1475	0.1715
Gaussian	0.1242	0.1494
Gaussian-linear	0.1308	0.1512

Table 2: Results with selection criteria MSE-LOO

Covariance and criteria	mean training+test MSE	predictive variance	mean mse + variance
Linear NLE	0.1774	24.0757	24.2531
Gaussian NLE	0.1271	23.5918	23.7189
Gaussian-linear NLE	0.1214	23.8945	24.0159
Linear MSE-LOO	0.1595	23.1274	23.2869
Gaussian MSE-LOO	0.1368	23.1295	23.2663
Gaussian-linear MSE-LOO	0.1410	23.3977	23.5387

Table 3: Training and test MSE vs prediction security (mean predictive variance)

In order to select a model, we compute the mean of training and test MSE, and the mean of predictive variance, see table 3. Considering the MSE, the best model is the GP with Gaussian-Linear covariance adapted with selection criteria NLE. Considering the predictive variance, the best model is the GP

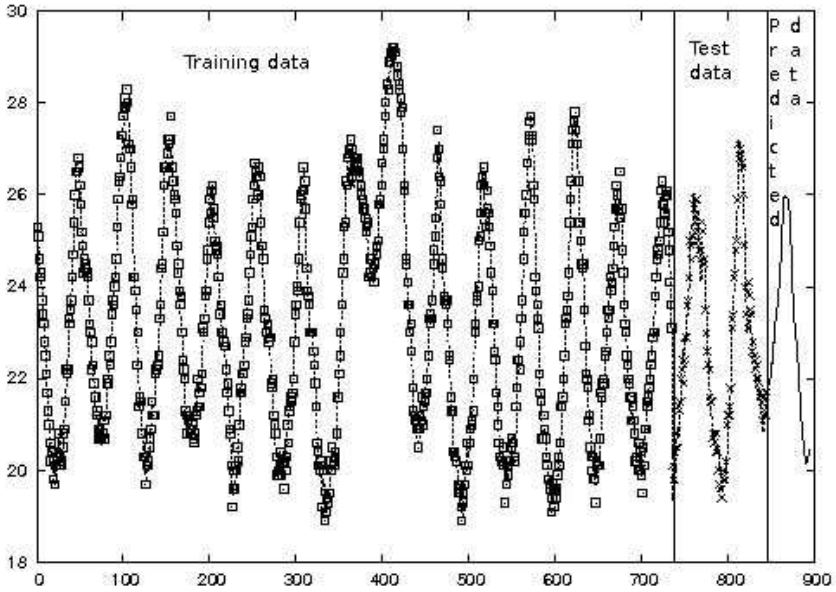

Fig. 1: Training, test and prediction: criteria NLE, covariance Gaussian-Linear

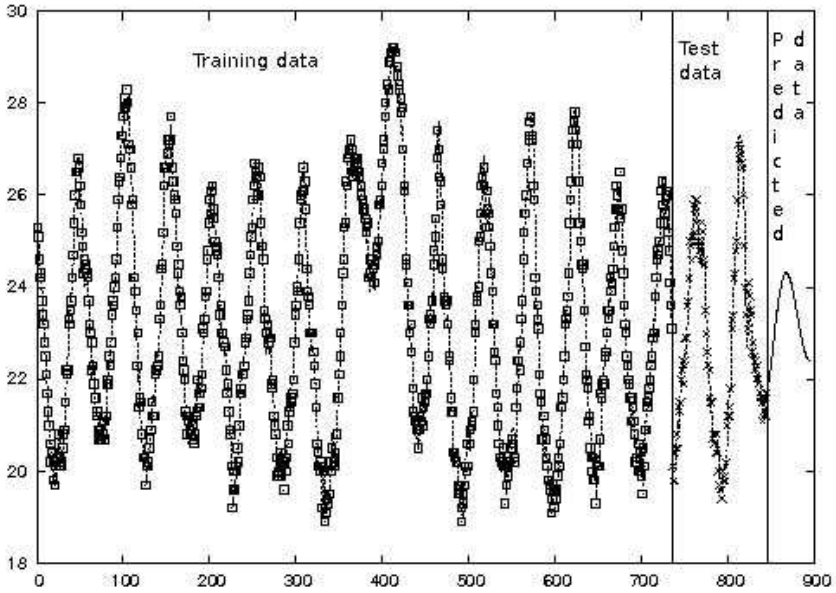

Fig. 2: Training, test and prediction: criteria MSE-LOO, covariance Linear

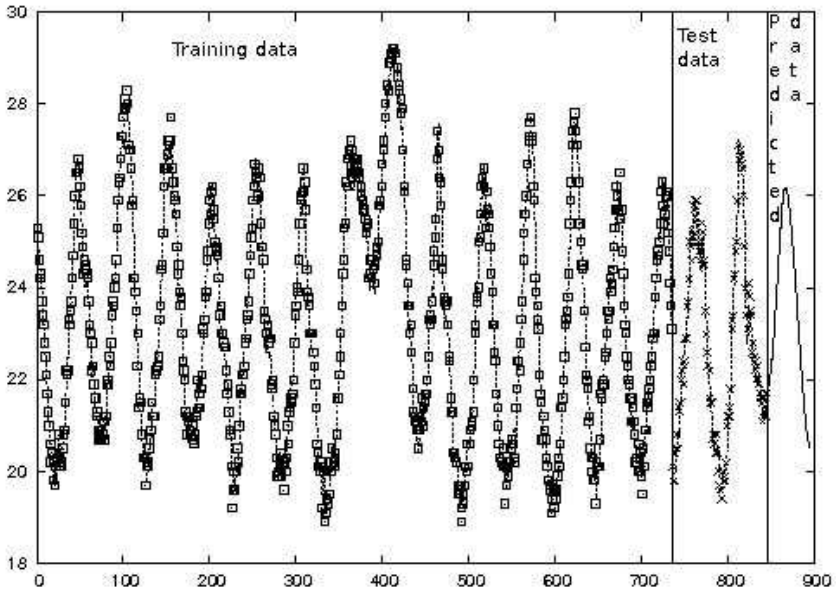

Fig. 3: Training, test and prediction: criteria MSE-LOO, covariance Gaussian

with Gaussian-Linear covariance adapted with selection criteria NLE. Finally, considering the sum of MSE and predictive variance, the best model is the GP with Gaussian covariance adapted with selection criteria MSE-LOO. As figures 1, 2 and 3 can show, predictive variance by itself seems not to be as informative as the MSE of training and test.

4 Conclusions

The main conclusion of this paper is that without good priors about the data generation process NLE leads to over-fitting (see table 1) and MSE-LOO becomes more reliable, as expected by Chapelle [8]. So the model chosen for competition is the GP with Gaussian covariance adapted with selection criteria MSE-LOO.

References

[1] B. Schölkopf and A. Smola. *Learning with Kernels*, MIT Press, Cambridge, MA, 2002.

[2] R. M. Neal. Bayesian learning for neural networks. *Lecture Notes in Statistics*, (118), 1996.

[3] S. Brahim-Belhouari, A. Bermak. Gaussian Process for nonstationary time series prediction. *Computational Statistics & Data Analysis*, 47 (4): 705-712 NOV 1 2004

[4] D. J. C. MacKay. Introduction to Gaussian Processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, NATO ASI Series, pages 133-166. Kluwer Academic Press, 1998.

[5] D. J. C. MacKay. Gaussian Processes: A replacement for supervised neural networks?. In *NIPS97 Tutorial*, 1997.

[6] C. K. I. Williams and C. E. Rasmussen, Gaussian Processes for regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Proceedings of Conference Advances in Neural Information Processing Systems*, NIPS, volume 8. MIT Press, 1996.

[7] R. M. Neal. Monte Carlo implementation of Gaussian Process models for Bayesian regression and classification. Technical Report, CRG-TR-97-2, Dept. of Computer Science, Univ. of Toronto, 1997.

[8] O. Chapelle. Some thoughts about Gaussian Processes. Nips Workshop on Gaussian Processes, 2005.

Bayesian Angle Information HMM with a von Mises Distribution and its Implementation using a Bayesian Monte Carlo Method

H. Sasaki[1], Y. Nakada[2], T. Kaburagi, and T. Matsumoto *

WASEDA University - Graduate School of Science and Engineering
55N-0403B 3-4-1 Okubo Shinjuku-ku Tokyo - Japan

Abstract. This paper proposes an angle information Hidden Markov Model (HMM) with a von Mises distribution within a Bayesian framework. To deal with angle information, this model expresses the emission probabilities of HMM using a mixture of von Mises distributions. Since closed form representation of posterior distribution is not available, this study implements Bayesian Monte Carlo methods to train parameters. The methods are used to compute posterior and related distributions associated with the proposed model. The scheme is applied to a pattern recognition problem of time series sequences. The proposed scheme improved the recognition rate by 3.1% over previous scheme.

1 Introduction

Angle information such as pen angles, joint angles, and direction angles are very important for several problems in pattern recognition, system control, signal processing and so on. In addition, such angle information usually involves noise or statistical uncertainty, which are observed in other information too. Therefore, we require a method that would enable us to deal with angle information that is combined with noise or statistical uncertainty. In order to deal with angle information in a probabilistic/statistical framework, one of the approaches is to utilize the von Mises distribution, which is a 'natural' distribution for angle information [1]. Recently, several studies have focused on the von Mises distribution and its extensions (e.g. von Mises Fisher distribution) to deal with angle or cyclic information in a natural manner (e.g. [2] [3]).

This paper proposes a novel semi-continuous HMM (continuous observation HMM) with von Mises distribution within a Bayesian framework for modelling time series data that contains angle information. In the proposed model, the emission probabilities are expressed by using a mixture of von Mises distributions. Although the computational costs of Bayesian Monte Carlo methods are high, these methods show several advantages, e.g. robustness of sparse and/or noisy datasets, and high precision computation in many models [4] [5]. Therefore, in order to train the parameters of the proposed HMM, this study has implemented a Bayesian Monte Carlo method based on [6], that has been successfully applied to several problems (e.g. [7] [8]). The proposed scheme has been demonstrated in an on-line handwriting character recognition problem that uses pen angle information.

*The authors would like to thank discussions with T. Kudo and K. Ito.

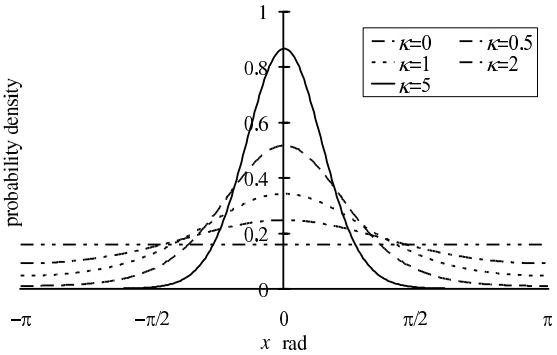

Fig. 1: Examples of a von Mises probability density function ($\mu = 0$).

2 The von Mises Distribution

Before describing the model specification, the von Mises distribution is presented. The von Mises distribution, also known as the circular normal distribution, is defined on the range $[-\pi, \pi)$ of the random angle information x. This distribution can be expressed as

$$\mathcal{M}(x; \mu, \kappa) := \frac{\exp\big(\kappa \cos(x - \mu)\big)}{2\pi I_0(\kappa)}, \tag{1}$$

where the mean direction parameter $\mu \in [-\pi, \pi)$ is the center of the distribution; the concentration parameter $\kappa \in (0, \infty)$, the sharpness of the distribution and $I_0(\kappa) := \sum_{j=0}^{\infty} \kappa^{2j} / 2^{2j} (j!)^2$, the modified Bessel function of the first kind and order 0 [9].

Figure 1 shows the shapes of the von Mises distribution with several settings of the concentration parameter κ. When κ reaches ∞, the von Mises distribution approximately becomes a normal distribution $\mathcal{N}(x; \mu, 1/\kappa)$ with a mean of μ and a variance of $1/\kappa$. On the other hand, when $\kappa = 0$, the von Mises distribution becomes a uniform distribution $\mathcal{U}(x; [-\pi, \pi))$ with the range $[-\pi, \pi)$ [1].

3 Bayesian Angle Information HMM

In this part, we describe a model specification of the Bayesian angle information HMM.

3.1 Observation data y and hidden variable z

(i) *Observation data* y

Firstly, consider an observation sequence $y := \{o_1, ..., o_T\}$, where T is the length of the sequence. In this paper, the observation data o_t comprises an angle information set v_{1t} and the other information set v_{2t} (if any), i.e.

$$o_t := (v_{1t}, v_{2t}), \quad t = 1, 2, ..., T. \tag{2}$$

More concretely, in the demonstration described in Sec. 5, v_{1t} represents pen angle information, and v_{2t} represents pen down/up information. For the sake of simplicity, in the following argument, we deal with both v_{1t} and v_{2t} as one-

dimensional values, and consider v_{2t} as a discrete value.

(*ii*) *Hidden variable z*
In a HMM framework, it is necessary to assume a hidden state sequence that underlies the observation sequence y. We denote this (discrete) hidden state sequence as $z := \{q_1, ..., q_T\}$, which has the same length as y. Here, $q_t \in \{s_1, .., s_N\}$; s_i represents the i-th state, and N represents the number of states.

3.2 HMM structure

The HMM structure depends on the HMM topology and the number of states N. There are two well-known topologies; 'ergodic' and 'left-to-right'. In the ergodic topology, state transition is permitted between any two states. On the other hand, in the left-to-right topology, state transition is restricted. More specifically, in many cases of the left-to-right topology, state transition is permitted from the current state $q_t = s_i$ to the same state $q_{t+1} = s_i$ or the next state $q_{t+1} = s_{i+1}$ where $t = 1, ..., T-1$ and $i = 1, ..., N-1$. The left-to-right topology is used in this study. In our formulation, which is described later, the main differences between these topologies can be found in the prior distributions of several parameters.

3.3 Likelihood function

Using the notations described above, the likelihood function, which denotes a conditional probability distribution for the observation sequence y given the parameter set θ, can be expressed as follows.

$$P(y \mid \theta) = \textstyle\sum_z P(y \mid \theta, z) P(z \mid \theta), \tag{3}$$

$$P(y \mid \theta, z) = \textstyle\prod_{t=1}^{T} P(v_{1t} | q_t, \theta) P(v_{2t} | q_t, \theta), \tag{4}$$

$$P(z \mid \theta) = P(q_1 | \theta) \textstyle\prod_{t=2}^{T} P(q_t | q_{t-1}, \theta), \tag{5}$$

where θ represents the parameter set. More details of θ and the probabilities in (3)–(5) can be described as follows.

(*i*) *Emission probability for angle information* v_{1t}
The emission probability for angle information v_{1t} is defined as a finite mixture of von Mises distributions:

$$P(v_{1t} | q_t = s_i, \theta) := \sum_{k=1}^{K} \nu_{ik} \mathcal{M}(v_{1t}; \mu_{ik}, \kappa_{ik}), \quad i = 1, ..., N, \tag{6}$$

where ν_{ik} is the mixture coefficient parameter of the k-th mixture in the state s_i, and ν_{ik} satisfies the constraints $\nu_{ik} \in [0, 1]$ and $\sum_{k=1}^{K} \nu_{ik} = 1$. $\mathcal{M}(\cdot; \mu_{ik}, \kappa_{ik})$ denotes the von Mises distribution (1) with the mean direction parameter μ_{ik} and the concentration parameter κ_{ik}.

(*ii*) *Emission probability for non-angle information* v_{2t}
The emission probability for discrete information v_{2t} is

$$P(v_{2t} = v_m | q_t = s_i, \theta) := b_{im}, \quad i = 1, ..., N, \;\; m = 1, ..., M, \tag{7}$$

where parameter b_{im} satisfies $b_{im} \in [0, 1]$ and $\sum_{i=1}^{N} b_{im} = 1$. M is the number of different symbols of v_{2t}.

(*iii*) *State transition probability for a hidden state* q_t
The state transition probability from $q_{t-1} = s_i$ to $q_t = s_j$ is defined as follows.

$$P(q_t = s_j | q_{t-1} = s_i, \theta) := a_{ij}, \quad i, j = 1, ..., N, \tag{8}$$

where $a_{ij} \in [0, 1]$ and $\sum_{j=1}^{N} a_{ij} = 1$.

(*iv*) *Initial state probability for an initial hidden state* q_1
The probability of initial state q_1 is defined by using parameter c_i.

$$P(q_1 = s_i | \theta) := c_i, \ \ i = 1, ..., N, \tag{9}$$

where $c_i \in [0, 1]$ and $\sum_{i=1}^{N} c_i = 1$.

3.4 Prior distributions for parameter set θ

Within a Bayesian framework, not only the likelihood function, but the prior distributions of the parameter set need to be defined. For the sake of simplicity, this study assumes that the parameters of the prior distribution are independent. Therefore, the prior distribution used in this study is defined as follows.

$$P(\theta) := \prod_{i=1}^{N} P(\nu_i) P(\mu_i) P(\kappa_i) P(b_i) P(a_i) \cdot P(c), \tag{10}$$

where $\nu_i := (\nu_{i1}, ..., \nu_{iK})$, $\mu_i := (\mu_{i1}, ..., \mu_{iK})$, $\kappa_i := (\kappa_{i1}, ..., \kappa_{iK})$, $b_i := (b_{i1}, ..., b_{iM})$, $a_i := (a_{i1}, ..., a_{iN})$, $c := (c_1, ..., c_N)$, and $\theta := (\{\mu_{ik}\}, \{\kappa_{ik}\}, \{\nu_i\}, \{b_i\}, \{a_i\}, c)$. For most of the prior distributions in (10), this study uses a 'natural conjugate prior' [4] [6], to simplify the implementation of the proposed model. The details of these prior distributions are described as follows.

(*i*) *Prior distributions for* ν_i, μ_i *and* κ_i
This part shows the settings of the prior distributions for the parameters ν_i, μ_i and κ_i, which correspond to the emission probability for angle information v_{1t}. For the mixture coefficient parameters ν_i, this paper defines the prior distribution $P(\nu_i)$ with a (natural conjugate) Dirichlet distribution, as shown in the following equation:

$$P(\nu_i) := \mathcal{D}(\nu_i; \zeta_i), \tag{11}$$

where $\zeta_i := (\zeta_{i1}, ..., \zeta_{iK})$ is the hyperparameter vector with $\zeta_{ik} > 0$ $(k = 1, ..., K)$, and $\mathcal{D}(\cdot; \zeta_i)$ denotes the Dirichlet distribution. In this paper, the direction parameters μ_i are fixed as $\mu_{ik} = 2\pi k/K - \pi$. [1] In other words, the prior distributions for μ_i can be described as follows.

$$P(\mu_i) := \prod_{k=1}^{K} \delta_{2\pi k/K - \pi}(\mu_{ik}), \tag{12}$$

where $\delta_x(y)$ is the Dirac delta function with the center x, i.e. $\delta_x(y) := \delta(y - x)$. For the concentration parameters κ_i, let the prior distribution $P(\kappa_i)$ be the gamma prior distribution with the constraint $\kappa_{i1} = \kappa_{i2} = \cdots = \kappa_{iK}$, i.e.

$$P(\kappa_i) := \mathcal{G}(\kappa_{i1}; \tau_i, \lambda_i) \cdot \prod_{k \neq 1} \delta_{\kappa_{i1}}(\kappa_{ik}), \tag{13}$$

[1] Actually, we also considered a uniform distribution $\mathcal{U}(\mu_{ik}; [-\pi, \pi))$ with range $[-\pi, \pi)$ for the direction parameters $\{\mu_i\}$. This uniform prior distribution, however, did not perform better than that of the fixed direction parameters in our preliminary experiment.

where $\mathcal{G}(\cdot;\tau_i,\lambda_i)$ is the gamma distribution with shape parameter τ_i and scale parameter λ_i.[2]

(*ii*) *Prior distributions for* b_i
For natural conjugation, let the prior distribution $P(b_i)$ be

$$P(b_i) := \mathcal{D}(b_i;\beta_i), \tag{14}$$

where $\beta_i := (\beta_{i1},...,\beta_{iK})$ represents the hyperparameter vector with $\beta_{ik} > 0$ $(k = 1,...,K)$.

(*iii*) *Prior distributions for* a_i
(*a*) *Ergodic topology:* The prior distribution $P(a_i)$ for the ergodic topology can be expressed by using the natural conjugate prior as

$$P(a_i) := \mathcal{D}(a_i;\alpha_i), \tag{15}$$

where $\alpha_i := (\alpha_{i1},...,\alpha_{iN})$ denotes the hyperparameter vector with $\alpha_{ij} > 0$ $(j = 1,...,N)$.
(*b*) *Left-to-right topology:* Considering natural conjugation of the prior distribution and the constraints of the left-to-right topology, the prior distribution $P(a_i)$ can be expressed as follows.

$$P(a_i) := \begin{cases} \delta_1(a_{ii}) \cdot \prod_{j\neq i} \delta_0(a_{ij}), & \text{if } i = N, \\ \mathcal{D}(a_i^*;\alpha_i^*) \cdot \prod_{j\neq i,i+1} \delta_0(a_{ij}), & \text{otherwise.} \end{cases} \tag{16}$$

Here, $a_i^* := \left(a_{ii}, a_{i(i+1)}\right)$, and $\alpha_i^* := (\alpha_{i1}^*, \alpha_{i2}^*)$ represents the hyperparameter vector with $\alpha_{i1}^*, \alpha_{i2}^* \geq 0$ $(i = 1,...,N-1)$.
(*iv*) *Prior distributions for* c
(*a*) *Ergodic topology:* By using the natural conjugation, let the prior distribution $P(c)$ for the ergodic topology be

$$P(c) := \mathcal{D}(c;\gamma), \tag{17}$$

where $\gamma := (\gamma_1,...,\gamma_N)$ is the hyperparameter vector with $\gamma_i > 0$ $(k = 1,...,N)$.

(*b*) *Left-to-right topology:* With the left-to-right topology presented in this paper, c_i is fixed as $c_1 = 1$ and $c_i = 0$ $(i = 2,...,N)$. Therefore, the prior distribution $P(c)$ in these cases can be described as

$$P(c) := \delta_1(c_1) \cdot \prod_{i\neq 1} \delta_0(c_i). \tag{18}$$

3.5 Setting hyperparameters

In this study, the hyperparameters are fixed. More specifically, in the demonstration of Sec. 5, all the components of the hyperparameters corresponding to the Dirichlet prior distributions are fixed as 1. [3] The other hyperparameters τ_i and λ_i are fixed as $\tau_i = 1$ and $\lambda_i = 50$.

[2] Although one can consider a gamma prior distribution $P(\kappa_i)$ without the constraint $\kappa_{i1} = \cdots = \kappa_{iK}$, this prior distribution did not perform well in our preliminary experiment.
[3] This is one of the simplest selections for the hyperparameters for the Dirichlet prior distribution.

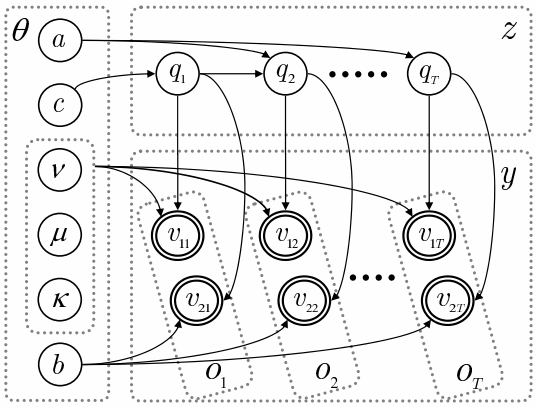

Fig. 2: Graphical representation of the Bayesian angle information HMM. Here, $a := \{a_i\}$, $b := \{b_i\}$, $\nu := \{\nu_i\}$, $\mu := \{\mu_i\}$ and $\kappa := \{\kappa_i\}$. The circles and double circles denote the unobserved variables and the observed variables, respectively. The arrows show probabilistic dependencies among variables. The dotted lines denote groups of variables. The hyperparameters and their probabilistic dependencies are omitted from this figure for visual clarity.

The proposed model described above can be summarized as shown in Fig. 2.

4 Implementation for Pattern Recognition Problems

4.1 Problem settings

Consider a pattern recognition problem as follows.

Problem:
Predict the unknown class $\mathcal{H}_{\text{NEW}}$ of a new observation sequence y_{NEW}, under the following conditions.
(i) Set of classes can be written as $\{\mathcal{H}_h\}_{h=1}^H$, i.e. $\mathcal{H}_{\text{NEW}} \in \{\mathcal{H}_h\}_{h=1}^H$,
(ii) Observation sequence set (training data) Y_h is given, for each class $\mathcal{H}_h (h = 1, ..., H)$, i.e. a one-to-one relation between Y_h and $\mathcal{H}_h$ exists.
(iii) Observation sequence set Y_h of the class $\mathcal{H}_h$ consists of several independent observation sequences, i.e. $Y_h := (y_{h1}, ..., y_{hL_h})$.

Here, y_{hl} denotes the l-th observation sequence y of class $\mathcal{H}_h$, and L_h is the number of observation sequences of class $\mathcal{H}_h$. H denotes the number of classes. In the demonstration in Sec. 5, $\mathcal{H}_h$ denotes the class based on characters, e.g. 'A', 'B', and so on.

4.2 An approach using the proposed model

An approach to solve this kind of problem is the scoring of the fitness between the new observation sequence y_{NEW} and the class $\mathcal{H}_h$ with its observation sequence set Y_h. Under the model specification described in the previous section, the fitness score between y_{NEW} and the class $\mathcal{H}_h$ can be expressed as follows:

$$\text{Score}(y_{\text{NEW}}; \mathcal{H}_h) := P(y_{\text{NEW}} \mid Y_h), \tag{19}$$

where $P(y_{\text{NEW}} \mid Y_h)$ represents the (conditional) marginal likelihood, which denotes the averaged likelihood function $P(y_{\text{NEW}} \mid \theta)$ of the posterior distribution $P(\theta \mid Y_h)$: [4]

$$P(y_{\text{NEW}} \mid Y_h) = \int P(y_{\text{NEW}} \mid \theta) P(\theta \mid Y_h) d\theta. \tag{20}$$

Here, the posterior distribution $P(\theta \mid Y_h)$ is derived from Bayes' theorem :

$$P(\theta \mid Y_h) = \frac{P(Y_h \mid \theta) P(\theta)}{\int P(Y_h \mid \theta) P(\theta) d\theta}, \quad P(Y_h \mid \theta) = \prod_{l=1}^{L_h} P(y_{hl} | \theta). \tag{21}$$

4.3 Implementation using Bayesian Monte Carlo methods

Unfortunately, there is no closed-form solution for the integration (20). Therefore, this paper utilizes Monte Carlo approximation methods:

$$P(y_{\text{NEW}} \mid Y_h) \approx \frac{1}{R} \sum_{r=1}^{R} P\Big(y_{\text{NEW}} \mid \theta^{(r)}\Big), \tag{22}$$

where $\{\theta^{(r)}\}_{r=1}^{R}$ denotes samples from the posterior distribution $P(\theta \mid Y_h)$. Here, in order to ease drawing samples $\{\theta^{(r)}\}_{r=1}^{R}$ using the Bayesian Monte Carlo method, we consider the joint posterior of the parameter set θ and the hidden state sequence set Z_h: [5]

$$P(\theta, Z_h \mid Y_h) = \frac{P(Y_h \mid Z_h, \theta) P(Z_h \mid \theta) P(\theta)}{\int \sum_{Z_h} P(Y_h \mid Z_h, \theta) P(Z_h \mid \theta) P(\theta) d\theta},$$

where

$$P(Y_h \mid Z_h, \theta) = \prod_{l=1}^{L_h} P(y_{hl} | z_{hl}, \theta), \quad P(Z_h \mid \theta) = \prod_{l=1}^{L_h} P(z_{hl} | \theta), \tag{23}$$

$Z_h := (z_{h1}, ..., z_{hL_h})$ represents the hidden state sequence set corresponding to the observation sequence set Y_h, and z_{hl} represents a hidden state sequence corresponding to the observation sequence y_{hl}. We obtain samples from $P(\theta, Z_h \mid Y_h)$, but not directly from $P(\theta \mid Y_h)$. Once we draw samples of (θ, Z_h) from the joint posterior $P(\theta, Z_h \mid Y_h)$, we can obtain samples of θ from $P(\theta \mid Y_h)$ by discarding samples of Z_h. [6] The details of our implementation can be summarized in Fig. 3.

5 Demonstration

In order to evaluate the proposed scheme, we applied it to an on-line handwriting character recognition problem. The handwriting-character database (HANDS-kuchibue_d-97-06) [10] was used as the dataset. The database contains six types of characters: Japanese Kanji (Chinese characters used in Japanese), Japanese

[4] Instead of the Bayesian integration approach (20), one can consider a maximum likelihood/posteriori approach. Unfortunately, several difficulties arise to deal with $\{\kappa_i\}$ in this approach.

[5] Considering the hidden state sequence set Z_h enables us to easily implement the Gibbs sampling method for several parameters [6].

[6] The two reasons for this are discussed in [6]. First, the Metropolis-Hastings procedures that average over Z_h are useful; however, these tend to perform poorly when the dimension of θ is large. Second, highly correlated elements of θ under $P(\theta \mid Y_h)$ are often nearly independent under $P(\theta, Z_h \mid Y_h)$; therefore, including Z_h in the sampling algorithm provides an expanded parameter space that may accelerate mixing with respect to θ.

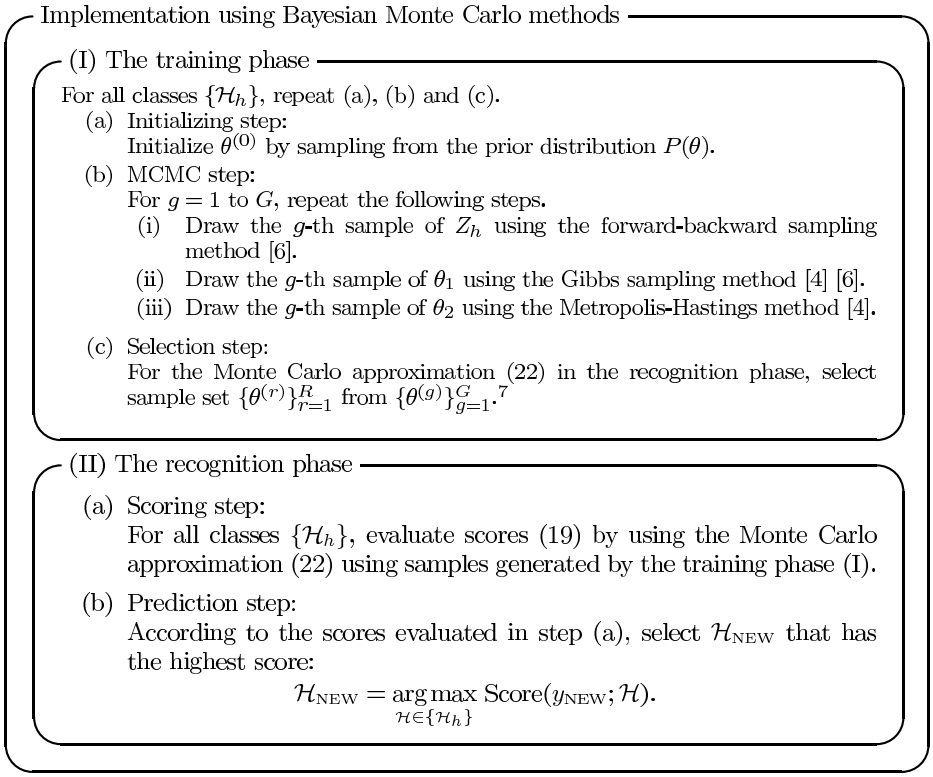

Fig. 3: Procedure of the pattern recognition problem. This procedure consists of two phases: (I) the training phase where we obtain samples of a parameter set θ and (II) the recognition phase where we estimate the scores for the classes $\{\mathcal{H}_h\}$. In this figure, let the parameter set $\theta := (\theta_1, \theta_2)$, where θ_1 can be dealt with by using the Gibbs sampling method. On the other hand, θ_2 cannot be dealt with by using this method. Therefore, the Metropolis-Hastings method is applied. Concretely, in the demonstration of this paper, θ_1 and θ_2 are $\theta_1 := (\{\nu_i\}, \{a_i\}, \{b_i\})$ and $\theta_2 := \{\kappa_i\}$, respectively.[8]

Hiragana, Japanese Katakana, Western alphabets, numerals and symbols (punctuation marks). For simplicity, this demonstration uses only 52 Western alphabets. The database comprises 120 datasets (120 persons) and each dataset comprises 116 characters of which 50 are upper cased and 66 are lower cased.

[7] With the Markov chain Monte Carlo (MCMC) method, we usually need to discard the samples in the first part [4]. In the demonstration of Sec. 5, we draw 500 samples during the MCMC step (I)(b) ($G = 500$), and we use the last 50 samples for the Monte Carlo approximation ($R = 50$) for each class.

[8] Note that the parameters $\{c_i\}$ are not required to be considered here because they can be fixed using the left-to-right topology of the implementation (See (18)). The parameters $\{\mu_i\}$ are also not required here, because this paper utilizes the fixed values for μ_i (See (12)).

Table 1: Recognition rates of alphabets both in lower case and upper case (Exp.1)

	1st [%]	top 3 [%]
Proposed	77.80	96.68
BDHMM[12]	74.14	95.60
DHMM[11]	74.66	96.16

Table 2: Recognition rates of upper cased characters (Exp.2)

	1st [%]	top 3 [%]
Proposed	96.40	98.10
BDHMM[12]	94.20	97.90
DHMM[11]	95.40	98.10

Table 3: Recognition rates of lower cased characters (Exp.3)

	1st [%]	top 3 [%]
Proposed	89.32	97.50
BDHMM[12]	87.80	96.82
DHMM[11]	87.12	97.05

Raw data of one character in this database consists of the two-dimensional pen position sequence and the pen down/up information sequence. From these sequences, we obtain the observation sequences $v_{1t} \in [-\pi, \pi)$ and $v_{2t} \in \{0, 1\}$, where v_{1t} represents the angle information and v_{2t} represents the pen down/up information.

The first 100 datasets (100 persons) were used for training, and the remaining 20 datasets (20 persons) were used for recognition. By using these training and test datasets, the following three experiments were conducted: Exp.1: train and recognize a total of 52 types of characters *both lower and upper case*; Exp.2: train and recognize 26 types of characters *only upper case*; and Exp.3: train and recognize 26 types of characters *only lower case*. [9]

For a comparison, two other approaches with HMMs were tested using the same datasets. The first is Bayesian discrete HMM (BDHMM) approach using Bayesian Monte Carlo methods reported in [12]. Bayesian Monte Carlo methods in [12] are also based on the forward-backward sampling reported in [6]. Emission probabilities of this approach are defined as multinomial distributions with quantized 16 directions, whereas the approach in this paper uses 16 mixture von Mises distributions. The second approach described in [11] uses discrete HMM (DHMM), which is not based on a Bayesian framework. This approach also uses 16 quantized directional observations. In this demonstration, the parameters in this approach have been set to the same values as those in [11]. Three approaches are evaluated using two criteria: (i) the first recognition rate, which is the rate

[9] In all of the above experiments, training was carried out using the following parameter (see Sec. 3.5 and 4.3): all components of the hyperparameters for the Dirichlet prior distributions are fixed as 1, the other hyperparameters are fixed as $\tau_i = 1$ and $\lambda_i = 50$. We also fixed the number of components $K = 16$, the number of MCMC steps $G = 500$, and the number of samples $R = 50$. The number of states N for each class is uniquely defined by using the method presented in [11]. This method decide N based on (i) the number of the angle information changes $|v_{1t} - v_{1(t+1)}|$ larger than an empirical threshold value $\phi_0 (> 0)$, and (ii) the number of the changes of the pen down/up information v_{2t}. In the demonstration of this paper, the maximal value of N was 18 and the minimal value was 4.

that the score of the correct character was ranked as first, and (ii) the top-three recognition rate, which is the rate that the score of the correct character was ranked in the top three. The first and the top-three recognition rates of three approaches are listed in tables 1–3.

From the experimental results shown in tables 1–3, one can observe the proposed scheme improved the first recognition rates in all three experiments, and the top three recognition rates in most of the experiments. Among three experiments, the largest improvement was observed in the first recognition rates of Exp.1; 3.1% over the scheme in [11] and 3.7% over [12].

6 Conclusion

This paper proposed a Bayesian continuous (observation) HMM with von Mises density function. Further, this paper applied a Bayesian Monte Carlo method to a pattern recognition problem by using the proposed model. The proposed scheme was applied to an on-line handwriting character recognition problem for evaluation. The scheme improved the first recognition rate by 3.1% over the scheme in [11].

References

[1] K.V. Mardia and P.E. Jupp, *Directional statistics*, Wiley, Chichester, 2000.

[2] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sura, Clustering on hyperspheres using Expectation Maximization. Technical Report TR-03-07, Department of Computer Sciences, University of Texas, Feb. 2003.

[3] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sura, Generative Model-based Clustering of Directional Data, The 9-th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2003.

[4] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, *Markov Chain Monte Carlo in Practice*, Chapman & Hall, London, 1996.

[5] R. Neal, *Bayesian Learning for Neural Networks*, Lecture Notes in Statistics, No. 118, Springer-Verlag, 1996.

[6] S. L. Scott, Bayesian Methods for Hidden Markov Models: Recursive Computing in the 21st Century, *J. Am. Stat. Assoc.*, vol.97, pp.337–351, 2002.

[7] D. Husmeier and G. McGuire, Detecting Recombination in 4-Taxa DNA Sequence Alignments with Bayesian Hidden Markov Models and Markov Chain Monte Carlo, *Mol. Biol. Evol.*, VSol.20, pp.315–337, 2003.

[8] S. Guha, Y. Li, and D. Neuberg, Bayesian Hidden Markov Modeling of Array CGH Data, *Harvard University Biostatistics Working Paper Series*, Working Paper 24, Jan. 4, 2006.

[9] G. N. Watson, *A Treatise on the Theory of Bessel Functions*, Cambridge University Press, 2nd edition, 1952.

[10] M. Nakagawa, TUAT Nakagawa Lab. HANDS-kuchibue_d-97-06, Tokyo University of Agriculture and Technology, 1996.

[11] H. Yasuda, K. Takahashi and T. Matsumoto, A Discrete HMM for On-line Handwriting Recognition, *Int. J. Pattern Recognit. Artif. Intell.*, vol.4, No.5, pp. 675–688, July 2000.

[12] A. Funada, H. Sasaki, Y. Nakada and T. Matsumoto, Monte Carlo HMM for On-line Handwriting Recognition, The 15th Annu. Conf. of the Japanese Neural Network Society 2005, pp.137–138, Sept. 2005 (in Japanese).

Bayesian Neural Networks for Time Series Prediction

Nariman Mahdavi ,M.B.Menhaj, Mehrdad Abedi

AmirKabir University of Technology - Dept. ofElectrical Engineering
424 Hafez Ave. -Tehran - Iran

Abstract. Conventional artificial neural networks (ANN) cannot produce a tolerable prediction result, especially when prediction of nonlinear time series corrupted with noise is the main concern. In this paper, we present a solution to this problem by applying Bayesian approach for learning process of multilayer feedforward neural networks (MLP). The Bayesian approach, in general requires explicit formulation of the underlying problem and conditioning on known quantities in order to draw inferences about unknown ones. In general, it is not obvious how large a time-window of inputs are suitable for accurate forecast, we encountered with a problem in which there are many possible input variables. In this paper we first use Automatic Relevance Determination (ARD) for selecting the regularization constants of each input and then by employing Markov Chain Monte Carlo (MCMC) technique, future values of time series are predicted.

1 Introduction

A time series can be modeled by

$$y_t = f(\underline{x}_t) + \varepsilon_t \quad ,t = 1,2,...,n$$

where $f(.)$ is an unknown function and $\underline{x}$ is a vector of lagged value of y, the amount of this time-lag is called the order of autoregressive model.

The determination of function f has been one of central topics in statistics for a long time. The ARIMA model [1] works well for linear time series but not adequate for nonlinear ones. Two popular nonlinear models are bilinear model [2] and threshold autoregressive model [3]. For these models the parameter estimation can be carried out by, for example, maximum likelihood function.

Although these models generally perform well, they have some limitations. First, without expertise it is possible to have false specification of the function form of the most suitable model. Second, the models themselves are limited and may not be able to capture some kinds of nonlinear behavior. To remedy these limitations, neural networks have been applied to modeling nonlinear time series by many authors, for example, Faraway and Chatfield [4]. While ANNs provide a great deal of promise, they also embody much uncertainty. Therefore, researchers are not certain about the effect of key factors on forecasting performance of ANNs (G. Zhang, et al [5]).

Usually a one-hidden-layer feedforward neural network can be suitably determined to represent the function $f(.)$, because this network with linear output units can approximate any continuous function arbitrarily well on a compact set by increasing the number of hidden units (Cybenko [6], Funahashi [7], Hornik, Stinchcombe and White, [8])

However, there is a question of weather the network weights corresponding to the best fitting network are unique? [9]. The answer is no. This led to the Bayesian learning of neural network which first proposed by MacKay [10] and further developed by Neal [11], Muller and Rios Insua [12], Holmes and Mallick [13], Freitas and Andrieu [14] to consider all possible weights by probability distribution.

A brief review on application and development of NNs to time series forecasting, with focusing on Bayesian Neural Networks (BNNs), is given by T. Zhang [15]. Geweke and Whiteman [16] present the principles of Bayesian forecasting, and describe recent advances in computational capabilities for applying them to economic forecasting. After that, Acernese and et al [17, 18] applied a hierarchical Bayesian learning scheme to time series identification with autoregressive NN models which overcomes the problem of identifying linear and nonlinear parts in time series during the training stage. In both papers, the Gaussian approximation of posterior pdf is used. Liang [19], proposed a Monte Carlo algorithm for BNN training with an application into time series. He used the indicator function for the selection of input variables which complicates the training process, because the separate control of each connection is required. In this paper, we employ the ARD to avoid unnecessary complications.

Some benefits of Bayesian methods are listed below [20, 21]:

1) The conventional training method of error minimization arises from a particular approximation to the Bayesian approach.
2) Regularization can be given a natural interpretation in the Bayesian framework.
3) Bayesian methods allow the value of regularization coefficients to be selected using only the training data without the need of validation data set.
4) It provides an objective framework for dealing with system complexity and preventing from over-parameterization.
5) Determining the relative importance of different inputs.

In this study, we apply BNNs to time series prediction by using MCMC, in association with the ARD technique for determining suitable regularization constant of each input.

The remaining part of this paper is organized as follows. In Section 2, we introduce the time series which we want to predict, followed by section 3 presents the Bayesian learning for MLP NNs and the ARD model. In Section 4, the results of conventional NN, BNN with ARD using Gaussian approximation, and BNN with ARD using MCMC are given along with a full discussion over the simulation results. Finally, section 5 concludes the paper.

2 Data characteristics

The methods in this paper will be tested on the time series which is used for prediction competition. Figure 1 shows the series. We suppose that the samples 1 through 787 are available and used for training the neural network and the goal of prediction is to find proper values of the next 50 samples. Then, the goodness of prediction is measured by evaluating its MSE between actual and predicted values as:

$$MSE1 = \frac{1}{15}\sum_{i=1}^{15}(y_i - \hat{y}_i)^2$$

$$MSE2 = \frac{1}{50}\sum_{i=1}^{50}(y_i - \hat{y}_i)^2$$

The MSE1 is measure of short-term forecast and MSE2 is for long-term.

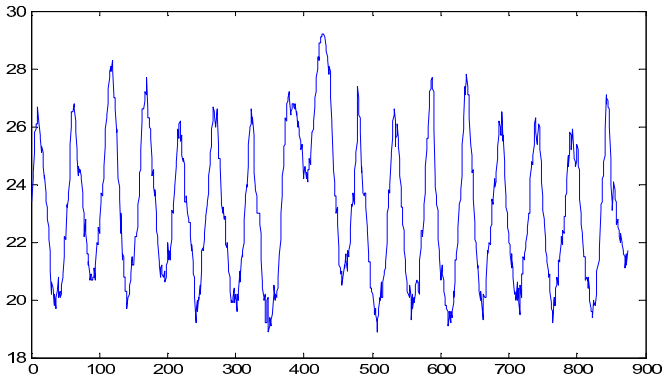

Fig. 1: View of time series

3 Bayesian learning for neural networks

In the first subsection, we describe the method of Hybrid Monte Carlo (HMC) which is useful for sampling from the multidimensional posterior probability density function of network parameters and in the second subsection we discuss the ARD model which selects the suitable regularization constants of each input.

3.1 Hybrid Monte Carlo

In training the network with a given architecture, the backpropagation approach (equivalent to maximum likelihood), finds a single best set weight values by minimization of a suitable error function. While in the Bayesian approach, the predictions are based on all possible values for the network parameters.

As pointed out in [22]-[23], the aim of Bayesian approach is to obtain the predictive distribution $P(y_{n+1}|\underline{x}_{n+1}, D)$ for a new test case $\underline{x}_{n+1}$, where y_{n+1} is the prediction for $\underline{x}_{n+1}$ and D is the set of "n" training data:

$$D = \{(\underline{x}_1, y_1), ..., (\underline{x}_n, y_n)\} \quad (1)$$

The predictive distribution can be modeled as follows:

$$P(y_{n+1}|\underline{x}_{n+1}, D) = \int_{R^N} P(y_{n+1}|\underline{x}_{n+1}, D, \underline{w})\, P(\underline{w}|D, \underline{x}_{n+1})\, d\underline{w} \quad (2)$$

where, $\underline{w}=[w_1,w_2,\cdots,w_N]$ is the vector of all neural network parameters (weights & biases) and "N" is its dimension. Since $\underline{x}_{n+1}$ has no effect on $P(\underline{w}|D,\underline{x}_{n+1})$, Eq. (2) can be rewritten as:

$$P(y_{n+1}|\underline{x}_{n+1},D)=\int_{R^N} P(y_{n+1}|\underline{x}_{n+1},D,\underline{w})\,P(\underline{w}|D)d\underline{w} \tag{3}$$

Clearly, this predictive distribution is a function of y_{n+1}. Therefore, for a single value prediction, we calculate the expected value of this distribution which is equal to its center of mass. If we consider a Gaussian distribution for $P(y_{n+1}|\underline{x}_{n+1},D,\underline{w})$ with mean $f(\underline{x}_{n+1},\underline{w})$ and variance σ^2, then we can write:

$$\hat{y}=E(P(y_{n+1}|\underline{x}_{n+1},D)=\int_{R^N} f(\underline{x}_{n+1},\underline{w})P(\underline{w}|D)d\underline{w} \tag{4}$$

where "*f*" is a mapping from input to output of a neural network with parameter $\underline{w}$.

For evaluating $\hat{y}$, we need at first to calculate the posterior probability density function:

$$P(\underline{w}|D)=\frac{P(\underline{w})P(D|\underline{w})}{P(D)} \tag{5}$$

where *P(*$\underline{w}$*)* is a prior pdf, chosen by designer. We select a Gaussian pdf with zero mean and large variance. It is important to note that the posterior pdf becomes more sharp by considering the data.

Now, the integral in (4) must be solved. It cannot be done analytically because of the high dimension of the weight space. We consider the Hybrid Monte Carlo (HMC) method [23], since it is faster than simple Monte Carlo or Gibbs sampling. The HMC method is a combination of the Metropolis Hasting and the stochastic dynamic method. A complete guide to MCMC methods is given by Andrieu and et al [24].

The Metropolis algorithm generates a sequence of vectors $\underline{w}_0,\underline{w}_1,\ldots$ that forms an ergodic Markov chain with stationary distribution *P(*$\underline{w}$*|D)*. After that, the integral is approximated by

$$\hat{y}\approx\frac{1}{M}\sum_{t=I}^{I+M-1} f(\underline{x}_{n+1},\underline{w}_t) \tag{6}$$

Here, *I* initial values are discarded and *M* values of "*f*" are averaged. In the HMC method, the candidate state $\underline{w}_t$ is generated according to gradient information. For this reason, the Hamiltonian function is defined as:

$$H(\underline{w},\underline{p})=E(\underline{w})+\frac{1}{2}|\underline{p}|^2 \tag{7}$$

where, P is the momentum vector and $E(\underline{w})$ is:

$$E(\underline{w})=\frac{|\underline{w}|^2}{2\omega^2}+\sum_{i=1}^{n}\frac{|y_i-f(\underline{x}_i,\underline{w})|^2}{2\sigma^2} \tag{8}$$

where, ω^2 is variance of the prior pdf which comes from the ARD model and it depends on which input component is more influential. The candidate state $(\underline{w}_i, \underline{p}_i)$ is found by solving the following equations:

$$\begin{cases} \dfrac{d\underline{w}}{d\tau} = \dfrac{\partial H}{\partial \underline{p}} = \underline{p} \\ \dfrac{d\underline{p}}{d\tau} = \dfrac{-\partial H}{\partial \underline{w}} = -\nabla E(\underline{w}) \end{cases} \tag{9}$$

Since the Hamiltonian dynamic method cannot be simulated exactly, Eq. (9) is discretized by step-size ε, using the leapfrog method:

$$\begin{aligned} \underline{p}(\tau + \frac{\varepsilon}{2}) &= \underline{p}(\tau) - \frac{\varepsilon}{2}\nabla E(\underline{w}(\tau)) \\ \underline{w}(\tau + \varepsilon) &= \underline{w}(\tau) + \varepsilon \underline{p}(\tau + \frac{\varepsilon}{2}) \\ \underline{p}(\tau + \varepsilon) &= \underline{p}(\tau + \frac{\varepsilon}{2}) - \frac{\varepsilon}{2}\nabla E(\underline{w}(\tau + \varepsilon)) \end{aligned} \tag{10}$$

with L leapfrog iterations, the candidate state is generated, and accepted with probability $\min\{1, \exp(-\Delta H)\}$. Hence, the Hamiltonian is minimized.

3.2 **Automatic Relevance Determination**

In the context of prediction, the selection of inputs which have the most effect on the output is very important. In the statistical method, a threshold is usually determined first according to the correlation between inputs and output and then the inputs which exceed the threshold value are selected and the rest are eliminated.

It is worth mentioning that it is not true to reject some inputs because of having correlationcoefficient only slightly less than the threshold. The correct way is to consider effects of all inputs simultaneously. In other words, the input which has only slightly greater effect contributes slightly more to the final prediction value. In the ARD model, this is done by adjusting the variance of the prior density function of each input. For example, if one input has a little effect on the output, the variances of the related weights become small, and leading a little contribution to the prediction value.

As we mentioned before, the predictive distribution (3) cannot be done analytically unless we assume Gaussian approximation for the posterior pdf, P($\underline{w}$|D) [10]. With such an assumption, the formulation of the ARD model is simple and given in [25] and is summarized below:

$$\gamma_c = k_c - \alpha_c Trace_c(A^{-1})$$

where γ_c is the number of well determined parameters in class c (each class can be viewed as the weights connected to each input), k_c is the total number of parameters in each class, α_c is the inverse of prior variance and A is the Hessian of error function when the weights have their most probable (MP) values obtained by the BP algorithm. Then, he re -estimation formulas for regularization constants are:

$$\alpha_c = \frac{\gamma_c}{\sum_{i \in c} (w_i^{MP})^2}$$

4 Time series prediction

In this section, we consider three types of predictions associated with Neural Networks. For all of these methods we require the normalized training data; this can be done by finding the maximum and minimum of time series and scaled down it to [-1,1]. The normalized time series are denoted by: $Y_1, Y_2, ..., Y_n$

The training pattern D is constructed as follows:

$$\underline{x}_i = \begin{bmatrix} Y_{i+k-1} \\ \vdots \\ Y_i \end{bmatrix}_{k \times 1}, \quad y_i = Y_{i+k} \quad , i = 1,2,...,n-k$$

Here, we use k previous values of time series for prediction of the next value. As a result, the total training patterns are n-k. The first test input is generated from this time series by choosing $i = n - k + 1$.

4.1 Conventional neural networks

We use a MLP neural network with 1 hidden layer and the BP algorithm as a learning rule to predict the future values of the time series. The number of neuron in hidden layer is 10 and the value of time lag is 8.

After predicting the first value $\hat{Y}_{n+1}$, the test inputs for the next predictions are built in the same way as before. The only difference is replacing the Y with $\hat{Y}$, everywhere we do not have the actual value of time series.

This prediction is very sensitive to the initial values of weights. Since the BP is a local search algorithm and the error function has more than one local minimum, it is possible that the algorithm converges to different local minimum depending on the initial weights. Figure 2 shows the best possible result.

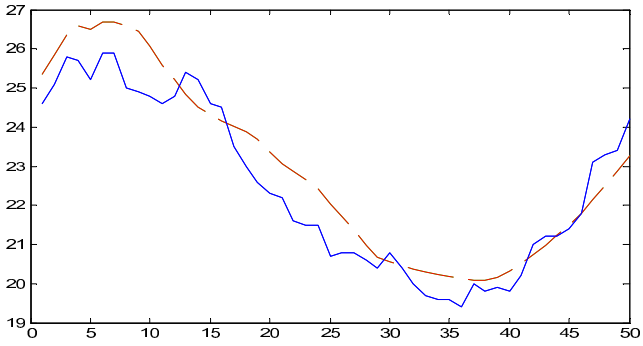

Fig. 2: NN prediction

The MSE1 value for the first 15 predictions is 0.9087 and the MSE2 value for all 50 predictions becomes 0.6004

4.2 BNN with ARD using Gaussian Approximation

Here, we apply the ARD model using Gaussian approximation to the same neural network considered in the previous subsection for determining proper regularization constant of each input. The simulation results show that the time-lags of 1,8,2,5 have the most important impact on the prediction, respectively. Figure 3 shows this prediction with the time lag of 8.

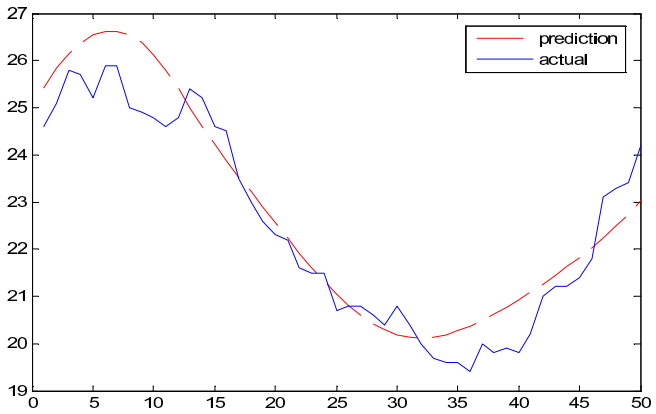

Fig. 3: BNN with ARD using Gaussian Approximation

The corresponding values of MSE1 and MSE2 are 0.8874 and 0.4820, respectively.

4.3 BNN with ARD using HMC

In the previous section, we determined the suitable variance weights, connected to each input by assuming Gaussian approximation for the posterior pdf of weights. We should note that this assumption is only true when the error function has only one peak. It would be better to use Monte Carlo sampling method for prediction as discussed in section 3 after the weights' variances are obtained.

The parameter values of the HMC method that we used for predictions are as follows: the initial value of prior variance is 100, the initial value of predictive distribution variance is 0.05, number of samples omitted at the start of chain (I) is 100, number of Monte Carlo samples returned (M) is 600, Leapfrog step-size (ε) is 0.005 and number of leapfrog iteration (L) is 10.

Now the input relevance is determined automatically by increasing the value of time-lag without getting worried about poor prediction, because the ARD mechanism assigns the right a-prior density function to each weight input. Figure 4 shows the prediction results for k=50.

The value of MSE1 and MSE2 are 0.2077 and 0.2054, respectively and as easily observed the Bayesian neural network improves the predictions in both short and long term.

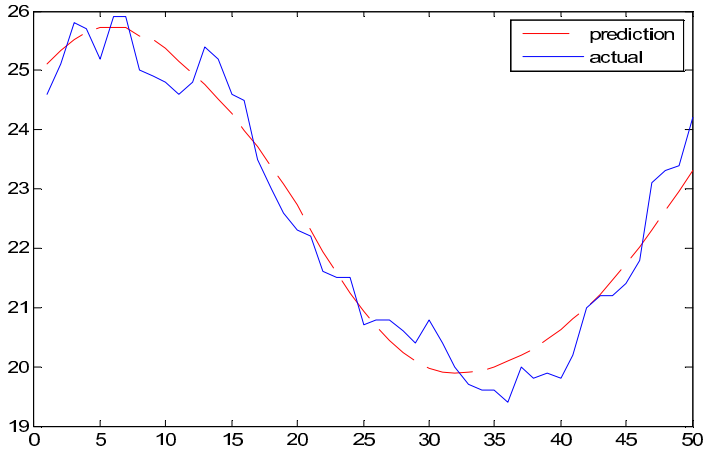

Fig. 4: prediction of BNN with ARD using HMC

5 Conclusion

In this paper, we proposed a BNN with Hybrid Monte Carlo method for time series prediction. The ARD technique was further used for determining suitable regularization constants of weights connected to each input. Therefore, if one input has a little effect on the output, the variances of the connected weights to that input become small. Consequently, these weights will have small values leading to a little contribution of the associated input to the final prediction. The simulation results approved that this combined method outperforms the conventional neural network and Bayesian learning with Gaussian approximation for predicting nonlinear time series.

References

[1] G.E.P. Box and G.M.Jenkins, Time Series Analysis, Forecast and control, holden day, san fransisco, 1970

[2] T.Subba Rao and M.M.Gabr, An introduction to Bispectral Analysis and Bilinear Time Series Models,Springer, NewYork, 1984

[3] H.Tong, Non linear Time Series: a Dynamical System Approach,Oxford University Press, Oxford, 1990

[4] J.Faraway and C.Chatfield, Time Series forecasting with neural networks: A comparative study using the airline data, Appl Statist.47: 231-250, 1998

[5] G.Zhang and BE.Patuwo, and M.Y. Hu, Forecasting with artificial neural networks: The state of the art, Int. J. Forecasting 14: 35-62 Elsevier, 1998.

[6] G.Cybenko, Approximation by superpositions of a sigmoidal function, Mathmatics of Control, Signals and System 2: 303-314, 1989

[7] K.Funahashi, on the pproximate realization of continuous mapping by neural networks, Neural Networks 2: 183-192, 1989

[8] K.Hornik and M.Stinchcombe and H.White, multilayer feedforward networks are universal approximator, Neural Networks 2: 359-366, 1989

[9] B.Curry and P.H.Morgan, Model Selection in Neural Networks: Some Difficulties, European Journal of Operational Research 170: 567-577, Elsevier, 2006

[10] D.J.C.Mackay, Bayesian Methods for Adaptive Models, Ph.D thesis, California Institute of technology, 1992.

[11] R.M.Neal, Bayesian Learning for Neural Network, Springer Verlag, NewYork, 1996

[12] P.Muller and D.R.Insua, Issues in Bayesian analysis of neural network models, neural computation 10: 749-770, 1998

[13] C.C.Holmes and B.K.Mallick, Bayesian radial basis functions of variable dimension, neural computation 10: 1217 1233, 1998

[14] N.Freitas and C.Andrieu, sequential Monte Carlo for Model Selection and Estimation of Neural Networks, ICASSPS, 2000

[15] T.Zhang and A.Fukushige, Forecasting Time Series by Bayesian Neural Networks, proceeding of IEEE conference, no 0-7803- 72786 , 2002

[16] J.Geweke and C.Whiteman, Bayesian Forecasting, Sep 2004. prepared for The handbook of Economic Forecasting, Amsterdam: North-Holland, 2006

[17] F.Acernese and R.de rosa and L.milano, A hierarchical Bayesian learning framework for autoregressive neural network modeling of time series, Proceedings of 3rd international Symposium on image and signal processing and Analysis, 2003

[18] F.Acernese, A.eleuteri, l.milano, A hierarchical Bayesian learning scheme for autoregressive neural network: application to the CATS benchmark, IEEE Proceedings, no 0-7803-83591, 2004

[19] F.Liang, Bayesian neural networks for nonlinear time series forecasting, Statistics and Computing 15: 13-29 Springer, 2005

[20] C.M.Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, 1995

[21] D.J.C.Mackay, Information Theory- Inference- and Learning Algorithms, Cambridge University Press, 2003

[22] R.Neal, Bayesian Training of Backpropagation Networks by the Hybrid Monte Carlo Method, Technical Report CRG-TR-92-1, Toronto, April 1992

[23] R.Neal, Probablistic Inference Using Markov Chain Monte Carlo Methods, Technical Report CRG-TR-93-1, Toronto, Sep 1993

[24] C.Andrieu and N.D.Freitas and A.Doucet and M.Jordan, An Introduction to MCMC for Machine Learning, Machine learning 50: 5-43, 2003

[25] D.J.C.Mackay, Probable Networks and Plausible Prediction- a review of practical Bayesian methods for supervised Neural Networks, computation in neural networks,1995

Multi-stage time series forecasting using Gaussian regression and generalised lags

Juan-Carlos Ibañez[1] and Granville Tunncliffe-Wilson[2] *

1- Neo Metrics Analytics - Dept. of Forecasting and Optimization
Arequipa 1, 2^{nd} floor, 28043 Madrid - Spain

2- Lancaster University - Dept. of Mathematics and Statistics
Lancaster LA1 4YF - United Kingdom

Abstract. We describe improvements in multi-step forecasting which are achieved by combining three strategies: the definition of states of past information using the generalised shift operator; non-linear function approximation using Gaussian regression; and multi-step prediction by recursive single-step construction. We illustrate the strategy using a well known data set.

1 Introduction

Linear statistical time series models are now well understood, and forecasting methods based on these models are well established [1]. However, for most real life processes to which these methods are applied, the linear model would be considered an over-simplification.

Unfortunately, the assumption that a process is driven by a nonlinear relation, results in forecasting becoming an exceedingly complex problem. Much effort has been put into the development of nonlinear methods and models for forecasting time series, as may be seen from the literature: [2], [3], [4] and [5].

We have carried out extensive experimentation using a range of methods for non-linear forecasting of real and simulated series. Our conclusions, [6], enable us to make clear recommendations regarding the construction of empirical non-linear multi-step forecasts of a stationary series. The key points of our strategy are:

a) the selection of the information set (states) from past observations, by application of the generalised shift operator, [7]. See §2.

b) the construction of non-linear approximating functions using a Kriging smoother, also called Gaussian Process Regression - closely related to the Support Vector Machine estimator - [8]. See §4.

c) use of the recursive scheme of [9] for construction of multi-step predictions. See §3.

The following three sections describe these points in turn, then in §5 we show how they are combined in our forecasting methodology. Finally, §6 presents the results of the application of our methodology to the classical Sunspot time series.

*The first author is grateful to the UK EPSRC and Unilever Research and Development for their partial support of this work.

2 Linear time series models with generalised predictors.

The most common linear model used in time series forecasting is the autoregressive (AR) model. Nonlinear models are generally extensions of the AR model, in that lagged values of the series are used as predictors. In either case it is necessary to select the number and lags of the predictors.

When developing a linear model, the problem reduces to selecting the order of the process. This can be done using the classical AIC or BIC criteria, [10, 11]. On the other hand, when developing a nonlinear model, the selection of the predictors is critical; *i.e.* nonparametric time series models are known to loose performance when too many predictors are used, due to the curse of dimensionality. The number of parameters in a linear model is equal to the dimension of the predictor space; whereas for nonlinear models the *effective* number of parameters grows much faster as a function of the dimension.

An approach which is more economical in its use of lagged values, consists in using as predictors linear combinations of the lagged series. Such a procedure has been used in the field of automatic control, where discrete Laguerre coefficients are used to define a finite set of p predictors, $(x_{0,t}, x_{1,t}, ..., x_{p-1,t})$. Each one being an infinite weighted sum of the present and past values of the process. The same type of predictors have been successfully used in speech processing [12, 13]. They were introduced in the time series field by [14] and [15, 16, 7]. The corresponding time series model can then be expressed as

$$x_{t+1} = \alpha_1 x_{0,t} + \alpha_2 x_{1,t} + ... + \alpha_p x_{p-1,t} + e_{t+1}, \tag{1}$$

with $e_t \sim \mathrm{WN}(0, \sigma_e^2)$.

The particular set of predictors which we use, though closely related to Laguerre coefficients, are defined in terms of the *generalised shift operator* W, i.e. the states $x_{k,t}$, $k = \{0, 1, ..., p\}$, at time t are given by

$$x_{k,t} = W^k x_t = W x_{k-1,t}, \tag{2}$$

where for $k = 0$, $x_{0,t} = x_t$ and W is defined in terms of the lag operator B and the *discount parameter* $\theta \in [0, 1]$ by

$$W = \frac{B - \theta}{1 - B\theta}. \tag{3}$$

We then call (1) the ZAR model, and the set of predictors the ZAR states. Note that for $\theta = 0$ the operator W coincides with the lag operator, $W = B$, and the ZAR model reduces to a pure AR. The appearance of the ZAR states is similar, at low frequencies, to the subset of series values shifted by multiples of $L = (1+\theta)/(1-\theta)$ lags, but, unlike this subet, the ZAR states form a complete basis of the past, so no information is lost by their use. Historically, W is closely related to the continuous time generalised shift operator used by Wiener. In practice, the states are computed via a recursive expression derived from (2) and (3):

$$x_{k,t} = \theta x_{k,t-1} + x_{k-1,t-1} - \theta x_{k-1,t}. \tag{4}$$

We now write (1) as the univariate ZAR model in *linear form*, as

$$x_t = \alpha_1 x_{0,t-1} + \alpha_2 x_{1,t-1} + ... + \alpha_p x_{p-1,t-1} + e_t \tag{5}$$

where $e_t \sim \text{WN}(0, \sigma_e^2)$. The model is therefore linear in the coefficients α_1, α_2, ..., α_p. It may also be shown that the ZAR(p) model can be written as an ARMA($p, p-1$) with prescribed moving average operator $(1-\theta B)^{p-1}$.

3 Multi Step Forecasting Tactics

The forecasting of time series further than one step ahead is not an easy problem even if the data generating process (DGP) is known. Once a time series model or a prediction method has been chosen, multi-step forecasts can be produced in several ways. We call these ways, *multi-step tactics* to distinguish them from optimal model-based prediction methods.

A multi-step tactic is therefore the form by which a multi-step forecast is obtained. Traditionally two basic tactics have been used: *plug-in* and *direct*. More recently a recursive tactic with promising results has been introduced [9]; following their terminology we call it *multi-stage*.

In the following, we describe these three tactics and give details of their properties. In each case, a function of predicting variables is used to construct a prediction, which is taken to be the *expected value* of the predicted variable, conditional on the predicting variables. The prediction error is therefore uncorrelated with the predicting variables; an assumption which is generally strengthened to independence. For simplicity of exposition assume that y_{t-1} is a sufficient summary of past information, and that the single-step predictor may be then be presented in the form of a DGP:

$$y_t = \phi(y_{t-1}) + \varepsilon_t, \tag{6}$$

with $\varepsilon_t \sim \text{WN}(0, \sigma_\varepsilon^2)$ and independent of $y_{t-s}, s \geq 0$.

First consider the tactic of *plug-in multi-step forecasting*, also known as *iterated model multi-step forecasting*. From the model (6), the one-step-ahead conditional expectation of the DGP is,

$$\text{E}\left[y_{t+1} \,|y_t\right] = \phi(y_t). \tag{7}$$

The 2-step-ahead forecast is then built successively from previous forecasts

$$\begin{aligned} \widehat{y}_{t+2|t} &= \text{E}\left[y_{t+2} \,|y_t\right] \\ &= \text{E}\left[\phi(\phi(y_t) + \varepsilon_{t+1}) \,|y_t\right], \end{aligned} \tag{8}$$

and similarly for any $k > 1$. The issue in (8) is the effect that the function $\phi(\cdot)$ has on the noise ε_{t+1}. Evaluation of such an effect can be done in several ways, as described in [3]:

- *Naïve*. It ignores the noise, replacing it by its mean value of zero.

$$\widehat{y}_{t+2|t} \quad = \quad \phi(\phi(y_t)) \tag{9}$$

  Its strength and weakness is its simplicity; it is easy and quick to compute but generally biased due to $\mathrm{E}\left[\phi(\mu+\varepsilon_{t+1})\right] \neq \phi(\mathrm{E}\left[\mu+\varepsilon_{t+1}\right]) = \phi(\mu)$, where $\mu = \phi(y_t)$.

- *Exact*, when the noise distribution, $\varepsilon_t \sim F_\varepsilon$, is known and usable.

$$\widehat{y}_{t+2|t} \quad = \quad \int \phi(\phi(y_t)+z) dF_\varepsilon(z) \tag{10}$$

  it is computationally complicated (requiring in general high dimensional integrals). It can be biased if F_ε is incorrectly selected.

- *Monte Carlo*, when the noise distribution $\varepsilon_t \sim F_\varepsilon$ is known but the integral above is not computable, it is possible to obtain a Monte Carlo forecast

$$\widehat{y}_{t+2|t} \quad = \quad \frac{1}{N}\sum_{i=1}^{N} \phi(\phi(y_t)+\varepsilon_t^{(i)}) \tag{11}$$

  where $\varepsilon_t^{(i)}$ are random samples $\varepsilon_t^{(i)} \sim F_\varepsilon$ for $i = 1, 2, .., N$, and N being the Monte Carlo sample size. This method is simpler than the previous but has the same difficulties: selecting the correct noise distribution.

- *Bootstrap*, which uses the empirical distribution of the estimation residuals $\widehat{\varepsilon}_t \sim \widehat{F}$:

$$\widehat{y}_{t+2|t} \quad = \quad \frac{1}{N}\sum_{i=1}^{N} \phi(\phi(y_t)+e^{(i)}) \tag{12}$$

  with $e^{(i)} \sim \widehat{F}$ for $i = 1, 2, .., N$; N being the bootstrap sample size. It is relatively easy to compute and should give good results if (7) is true, but only if the errors are truly independent and stationary, *e.g.* not subject to conditional heteroskedasticity.

The second tactic is *direct multi-step forecasting*. This tactic tries to solve the multi-step forecasting problem directly by modelling the conditional expectations:

$$\mathrm{E}\left[y_{t+k}\,|y_t\right] = \phi_k^{\mathrm{D}}\left(y_t\right), \quad k > 0 \tag{13}$$

Therefore for each desired step k, a direct modelling of the relation between y_{t+k} and y_t is necessary. The forecast is obtained by simple evaluation of ϕ_k^{D}. Two clear disadvantages are: it involves fitting a new predictor for each step and the residuals are generally autocorrelated. Construction of the predictor is therefore not statistically efficient. However, a well constructed direct multi-step predictor

should be unbiased, overcoming one of the possible draw-backs of the plug-in multi-step predictor.

The third tactic is *multi-stage multi-step forecasting*. This was introduced in time series by [9]. Similarly to the direct method, it constructs each conditional expectation

$$E\left[y_{t+k}\left|y_t\right.\right] \quad = \quad \phi_k^{\mathrm{MS}}\left(y_t\right), \quad k>0$$

but it is based on a sequence of recursive steps. For example under model (6)

$$\begin{aligned} E\left[y_{t+2}\left|y_t\right.\right] &= E\left\{E(y_{t+2}\left|y_{t+1}\right.)\left|y_t\right.\right\} \\ &= E\left\{\phi(y_{t+1})\left|y_t\right.\right\} \qquad (14) \\ &= \phi_2^{\mathrm{MS}}(y_t). \qquad (15) \end{aligned}$$

Thus at step k the predictor $\phi_k^{\mathrm{MS}}(y_t)$ is constructed by modelling the pairs $(\phi_{k-1}^{\mathrm{MS}}(y_{t+1}) \quad , \quad y_t)$ rather than $(y_{t+k} \quad , \quad y_t)$ which are modelled in the direct tactic.

This tactic shares one of the disadvantages of the direct method, that a different functional relationship has to be fitted for each k. However, at each of the recursive steps y_t is used to predict a *function* of y_{t+1}, so is efficient. In general, the predictor set needs to be large enough to assure (14) holds. As the authors in [9] show, their "multi-stage nonparametric forecast has smaller asymptotic mean-squared error than the direct nonparametric". It is easy to show that, for linear models, multi-stage and plug-in forecasts coincide.

The practical estimation of the functions ϕ_k^{MS} involves two steps, first a forecasting of the in-sample data using the previous function ϕ_{k-1}^{MS}, and then a model fitting of the pairs $\left(\phi_{k-1}^{\mathrm{MS}}(y_{t+1}), y_t\right)$. The authors in [9] found that the application of this multistage tactic with nonparametric state smoothing has problems of over-smoothing as k grows, because the kernel smoother which they use has a tendency to "flatten" the functional relationship at each step. An important modification which we propose is to use a Kriging smoother which is much less sensitive to this problem. We describe this in the next section.

4 Kriging/Gaussian Process Prediction

The simplest form of Gaussian Process (GP) modelling, represents a set of responses $y^{(i)}$ as a function of explanatory variables $\mathbf{x}^{(i)} \in \mathfrak{X} \subseteq \Re^p$, by

$$y^{(i)} = G(\mathbf{x}^{(i)}) + e^{(i)}, \quad i = 1, 2, ..., N \qquad (16)$$

or in compact form

$$\mathbf{y} = G(\mathbf{x}) + e, \qquad (17)$$

where $e^{(i)}$ are white noise measurement errors with variance τ^2. The key to construction of the smooth approximating function $G(\mathbf{x})$ is the assumption that it is the realisation of a zero-mean Gaussian random field with a specified covariance kernel. (If the mean is not assumed to be zero, we apply a suitable correction

to make it so.) This just means that the values of $G(\mathbf{x}^{(i)})$ at any finite set of points $\mathbf{x}^{(i)}$ have a jointly normal distribution with mean zero and covariances $K_{ij} = K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$.

Given that the values of $\mathbf{y}$ have been observed, the value of $G(\mathbf{x})$ at *any* point $\mathbf{x}^{(\text{new})}$ can be estimated, as described below, as a conditional expectation, evaluated by solving linear equations. From the model (16) we can write

$$\mathbf{y}|X \sim N(0, K + \tau^2 I), \tag{18}$$

where K has elements K_{ij}, and is usually expressed as $K = \sigma^2 R$, where R is a correlation matrix. From the context of spatial statistics, the measurement error is called the nugget effect and τ^2 is the nugget effect variance.

It is well known that the best predictor of $G(\mathbf{x}^{(\text{new})})$, at any point $\mathbf{x}^{(\text{new})}$, *i.e.* that which minimises its mean square error, is its conditional expectation given the data. To construct this we also need the vector k whose elements k_i are the covariances between $G(\mathbf{x}^{(\text{new})})$ and $G(\mathbf{x}^{(i)})$. Then, with k' the transpose of k,

$$E[G(\mathbf{x}^{(\text{new})})|\mathbf{y}] = k' \left(K + \tau^2 I\right)^{-1} \mathbf{y}, \tag{19}$$

The accuracy of the estimate may also be determined as the conditional variance:

$$\text{Var}[G(\mathbf{x}^{(\text{new})})|\mathbf{y}] = \text{Var}[G(\mathbf{x}^{(\text{new})})] - k' \left(K + \tau^2 I\right)^{-1} k. \tag{20}$$

These formulae follow from the application of Bayes Law, and the Bayes Linear approach [17] proposes the same solution from a slightly different position. In complicated problems where the inclusion of a priori information is needed or desired, it is easier to specify the prior for some parameters by their expectation and covariance rather than using a full prior probability function. Therefore this approach can be seen as a simpler definition of the prior specifications.

To fully determine the Gaussian process we need to specify the parametric form of the covariance $K(\mathbf{z}, \mathbf{x})$ between the process at any two points $\mathbf{z}$ and $\mathbf{x}$. In our applications we have used the kernels:

- Gaussian kernel,
$$K(\mathbf{x}, \mathbf{z}) = \exp\left(-s\|\mathbf{x} - \mathbf{z}\|_2^2\right). \tag{21}$$
- Hybrid Gaussian-Linear kernel
$$K(\mathbf{x}, \mathbf{z}) = \alpha_1(1 + \mathbf{x}^T\mathbf{z}) + \exp\left(-s\|\mathbf{x} - \mathbf{z}\|_2^2\right),$$
with $\alpha_1 \geq 0$.

In both of these the term Gaussian is used to describe the functional form of the covariance kernel by analogy with the Gaussian density function, but this is not required for the application of the Gaussian Process smoother.

In practice we need to determine the *hyperparameters*: s, which is the scale factor for the Gaussian kernel; $\lambda = \tau^2/\sigma^2$, which is the noise to signal ratio of the observations; and, if appropriate, α_1, which is the shrinkage factor for the linear part of the hybrid kernel. We determine these by a search for the values which yield the lowest cross-validation sum of squares of prediction errors, for the given observations.

5 Combined Treatment: Multi-Stage Time Series Forecasting using Gaussian Regression and Generalised Lags

We propose the combination of the three tools described above: ZAR linear states as predictors, the Multi-Stage tactic for forecasting, and Kriging (Gaussian Process Regression) approximation of non-linear functions. The use of ZAR states as defined in §2, has the main advantage that a smaller number of predictors may used; the multi-stage tactic retains statistical efficiency at high lead times, and Kriging regression avoids the search problems of Neural Networks, but suffers little from the curse of dimension associated with non-parametric regression. We refer to our model as Krige(ZAR), when using the simple Gaussian kernel, and KrigeLin(ZAR) when using the hybrid Gaussian-Linear kernel.

We now describe the steps to produce the multi-step forecast $\hat{x}_{T+k}$, given the series x_t for $t = 1, 2, \ldots, T$. We first determine a suitable ZAR(θ, p) base-model (see §5.1), then:

1. Compute the ZAR(θ, p) states: $Z_t = (z_{0,t}, z_{1,t}, \ldots, z_{p-1,t})$ for $t = 1, 2, \ldots, T$.
2. 1-step-ahead prediction:
   (a) Estimate the hyperparameters $h^{(1)} = (\alpha, s, \lambda)$ for the Kriging predictor with responses x_{t+1} and regressors Z_t, for $t = 1, 2, \ldots, T-1$.
   (b) Produce the desired forecast $\hat{x}_{T+1}$ using this Kriging predictor applied to the state Z_T.
3. k-step-ahead prediction (applied from $k = 2$ up to the maximum lead time):
   (a) Produce $(k-1)$-step-ahead predictions for the sample data under the previous Kriging model: $\hat{x}^{(k-1)}_{t+k-1}$, for $t = 1, 2, \ldots, T-k+1$; the superscript indicates the forecasting lead time of $k-1$.
   (b) Estimate the hyperparameters $h^{(k)} = (\alpha, s, \lambda)$ for the Kriging predictor with responses $\hat{x}^{(k-1)}_{t+k}$ and regressors Z_t for $t = 1, 2, \ldots, T-k$. That is, we use the previous $(k-1)$-step predictions of *k-step-ahead values* as targets.
   (c) Produce the desired forecast $\hat{x}_{T+k}$ using this Kriging predictor applied to the state Z_T

Now that the multi-step forecasting algorithm has been described, we explain the procedure used to select the ZAR base model which it uses.

5.1 Selection of the base-model

The goal is to select the best set of predictors to be used as regressors in Kriging Regression. Following our methodology of using ZAR base-models, the problem reduces to choosing θ and p. We propose the following two identification steps:

1. **Selection of the "Region" of best linear ZAR(θ, p) models.** Compute Schwarz's (BIC) Information criteria for the ZAR(θ, p) models with $\theta \in \{0, 0.1, ..., 0.9\}$ and $p \in \{1, 2, ..., p_{\max}\}$, where $p_{\max}$ must be chosen sufficiently large that the BIC has a well defined minimum at some lower order. Plot the BIC contours plots with coordinates (θ, p) and choose a contoured region, R_{ZAR}, around the minimum AIC values.

2. **Selection of the best ZAR(θ, p) base model for Kriging.** For each of the base-models in R_{ZAR}, define the set of predictors as the ZAR states at time t, $(z_{0,t}, z_{1,t}, ..., z_{p-1,t})$ and the 'predicting variable' by $z_{0,t+1}$ (i.e. one step ahead forecast giving the ZAR states). Then fit a Kriging model using GCV and save its Mean Squared Generalised Cross Validation Error. Select the model with smallest GCV.

This procedure was derived from experience gained after performing a large scale assessment with real and simulated time series, see [6] for details.

6 A classical Example: the Sunspot series

In this section we assess the performance of our multi-step forecasting methodology with a real dataset. We consider a popular and challenging real time series: the annual Sunspot Numbers (1700-1955). This dataset was chosen because of its intensive modelling in the literature. In particular, we are comparing our tools with the nonlinear time series methods reported in [18] and more recently with the Bayesian Neural Networks proposed in [19].

In [18], the authors proposed the estimation of subset Bilinear models; they compared the multi-step forecast performance of various models: AR, subset AR, SETAR and subset Bilinear. In [19] the same datasets were assessed, and a Bayesian methodology for Neural Network estimation of time series was presented. The models the authors considered were: Bayesian Neural Network, Traditional Neural Network and Nonparametric Regression.

The forecasting methods that we assess are:

- ZAR(θ, p).The Linear ZAR time series model, with plug-in forecasting tactic,
- Krige(ZAR(θ, p)). The Nonlinear Kriging Regression with Gaussian kernel and ZAR base-models with multi-stage forecasting tactic,
- KrigeLin(ZAR(θ, p)). The Nonlinear Kriging Regression with Linear-Gaussian kernel and ZAR base-models with multi-stage forecasting tactic.

We follow [18] and split the 256 observations into two sets: the first 221 are used for model estimation, while the out-of-sample prediction is calculated with forecasts of the last 35 observations. Forecasts are produced for lead-times 1 to 6.

We applied the modelling procedures described above. The best ZAR linear model according to the BIC is ZAR(0.3, 6) and the selected region R_{ZAR} is the

one determined by $R_{\text{ZAR}} = \{(p,\theta) : p \in \{3,4,5,6\}, \theta \in [0.15, 0.8]\}$. The base model in R_{ZAR}, chosen with the smallest GCVS for Kriging regression, was ZAR$(0.7, 5)$.

Summarising, our selected forecasting methods are to use the linear ZAR$(0.3, 6)$, Krige(ZAR$(0.7, 5)$) and KrigeLin(ZAR$(0.7, 5)$) models. Table 1 summarise previous results from [18] and [19] through the Mean Squared Prediction Error for lead times $h = 1, 2, .., 6$ (MSE(h)). For comparison, the results for our selected models are presented in Table 2.

Summary of previous performances on the multi-step forecasting of the Sunspot series [19]							
Model	AR(9)	Subset AR	SETAR$(2, 4, t-12)$	Subset Bilinear	BNN	TNN	Kernel
in-sample MSE	199.27	203.21	153.71	124.33	124.74	162.94	78.51
MSE(1)	190.89	214.1	148.205	123.77	142.03	171.99	76.8
MSE(2)	414.83	421.4	383.9	337.54	347.85	407.71	317.3
MSE(3)	652.21	660.38	675.59	569.79	509.60	607.18	549.14
MSE(4)	725.85	716.08	773.51	659.047	482.21	615.52	779.73
MSE(5)	771.04	756.39	784.27	718.866	470.35	617.24	780.16
MSE(6)	-	-	-	-	468.18	578.15	736.03

Table 1: MSPE(h) for the Sunspot series, as reported in [18] and [19]. Description of the models: AR(9), autoregressive with order 9; subset AR, autoregressive with lags: $1, 2, 9$; SETAR$(2, 4, t-12)$, Self-Extracting Threshold Autoregressive with 2 regimes (orders 2 and 4) and threshold variable $t-12$; Subset Bilinear, Bilinear with lags $1, 2, 3, 4, 6, 9$; BNN, standard Bayesian Neural Networks using lags 1 to 9; TNN, (Reversible Jump) Bayesian Neural Networks using lags 1 to 9; Kernel, Nonparametric Regression using lags 1 to 3

Our findings are that multi-step Kriging forecasting with Multi-Stage tactic achieves better or similar results to the best models reported in the literature, which are Bayesian Neural Networks and Bilinear models.

Multi-step Forecasting of the Sunspot series using the Proposed Tools			
MODEL	ZAR$(0.3, 6)$	Krg(ZAR$((0.7, 5))$	KrgLin(ZAR$((0.7, 5))$
TACTIC	plug-in	multi-stage	multi-stage
in-sample MSE	231.04	92.74	92.63
MSE(1)	194.37	149.91	150.05
MSE(2)	392.29	337.91	337.19
MSE(3)	585.22	464.77	464.55
MSE(4)	608.38	480.56	480.30
MSE(5)	613.05	455.08	454.72
MSE(6)	616.45	432.18	433.98

Table 2: MSPE(h) for the Sunspot series using ZAR$(0.3, 6)$, Krige(ZAR$(0.7, 5)$) and KrigeLin(ZAR$(0.7, 5)$).

The Multi-stage Kriging forecasting methodology outperforms all the other techniques at higher lead times. Low lead time forecasts obtained with this methodology are also very competitive. We conclude from these comparisons that the procedures we have described can be highly commended for non-linear time series forecasting.

References

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

[2] H. Tong. *Non-linear Time Series: A Dynamical System Approach*, volume 6 of *Oxford Statistical Science Series*. Oxford University Press, Oxford, 1990.

[3] C. W. J. Granger and T. Teräsvirta. *Modelling Nonlinear Economics Relationships*. Oxford University Press, 1993.

[4] P. H. Franses and D. van Dijk. *Non-linear time series models in empirical finance*. Cambridge University Press, 2000.

[5] J. Fan and Q. Yao. *Nonlinear Time Series: Nonparametric and Parametric Methods*. Springer, 2004.

[6] J. C. Ibañez. New tools for multi-step forecasting of nonlinear time series. *PhD Thesis. Lancaster University, U.K.*, 2006.

[7] A. S. Morton and G. Tunncliffe-Wilson. A class of modified high order autoregressive models with improved resolution of low frequency cycles. *Journal of Time Series Analysis*, 25:235–250, 2004.

[8] C.K.I. Williams and C.E. Rasmussen. Gaussian processes for regression. In D. S. Touretzky M. C. Mozer and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, pages 514–520. The MIT Press, 1996.

[9] R. Chen, L. Yang, and C. Hafner. Nonparametric multistep-ahead prediction in time series analysis. *Journal of the Royal Statistical Society B*, 66(3):669–686, 2004.

[10] H. Akaike. A new look at the statistical model identification. *IEEE Transactions of Automatic Control*, AC-19:716–723, 1974.

[11] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[12] H. W. Strube. Linear prediction on warped frequency scale. *J. Acoust. Soc. Am.*, 68-4:1071–1076, 1980.

[13] A. Härmä, M. Karjalainen, V. Välimäki, L. Savioja, U. Laine, and J. Huopaniemi. Frecuency-warped signal processing for audio applications. *J. Audio Eng. Soc.*, 48-11, 2000.

[14] B. Wahlberg and E.J. Hannan. Parametric signal modelling using laguerre filters. *The Annals of Applied Probability*, 3:467–496, 1993.

[15] A. S. Morton. Spectral analysis of irregularly sampled time series data using continuous autoregressions. *PhD Thesis. Lancaster University, U.K.*, 2000.

[16] A. S. Morton and G. Tunncliffe-Wilson. Forecasting economic cycles using modified autoregressions. In *Proceedings of the Business and Economic Statistics Section, Presented at the Joint Statistical Meetings*, pages 207–212, Indianapolis, Aug. 2001. American Statistical Association.

[17] M. Goldstein. Bayes linear analysis. *Encyclopaedia of Statistical Sciences, Wiley*, 3:29–34, 1999.

[18] M. M. Gabr and T. Subba-Rao. The estimation and prediction of subset bilinear time series models with applications. *Journal of Time Series Analysis*, 2(3):155–171, 1981.

[19] F. Liang. Bayesian neural networks for nonlinear time series forecasting. *Statistics and Computing*, 15:13–20, 2005.

Multistep-ahead time series prediction by nearest neighbor based method

Syed Rahat Abbas and Muhammad Arif *

Pakistan Institute of Engineering and Applied Sciences (PIEAS),
Department of Computer and Information Sciences, Islamabad, Pakistan

Abstract. In this paper, a modified nearest neighbor based method has been proposed that can be used for multistep-ahead time series prediction. In the original time series, optimal selection of embedding dimension that can unfold the dynamics of the system is improved by using upsampling of the time series. Zeroth order normalized cross-correlation, which is amplitude invariant, is used instead of commonly used Euclidian distance as a selection criterion for nearest neighbor. The proposed method is used to predict the ESTSP competition time series.

1 Introduction

In time series prediction, the past values of a time series are utilized to predict its future values. A variety of prediction methods ranging from statistical methods to artificial intelligence based methods are being used. Nearest neighbor method is a pattern matching method in which the most recent pattern of the time series (reference pattern) is matched with all the past patterns (candidate patterns). The prediction is carried out by the next value of the best matched pattern.

Nearest neighbor method, initially proposed by Cover and Hart [1]. Its variants have been used for classification, pattern recognition and prediction. Modifications in the nearest neighbor method were suggested time to time for various prediction problems. Time series prediction using delay coordinate embedding was proposed in [2]; the mixture of direct and iterated method for prediction using four nearest neighbors with interpolation was carried out and method was applied for Santa Fe time series prediction competition in 1992. The comparison of nearest neighbor method with other method for prediction of foreign exchange shows that results are data dependent [3]. Simultaneous nearest neighbor method performed marginally better than ARIMA and random walk methods as reported in [4]. The prediction of chaotic behavior of market response is carried out using multivariate nearest neighbor method for precise prediction [5]. Divide and conquer approach to develop pair-wise class nearest neighbor method was proposed in [6]. Locally adaptive metric nearest-neighbor classification method was also proposed in [7]; they have used updating of weighted distance for getting optimal nearest neighbors. It was proposed that advanced data structures significantly reduce the execution time of nearest neighbor regression [8]. Subset features space was used by to improve nearest neighbor classification [9]. Subspace of candidate was used by [10] for fast search of nearest neighbors. Discriminate adaptive nearest neighbor classification was suggested by

* This work is financially supported by Indigenous PhD program of higher education commission (HEC), of Pakistan.

[11]. They used local linear discriminant analysis to estimate an effective metric for computing neighborhoods.

In this work we have used nearest neighbor method with upsampling of time series before embedding and it is also proposed that zeroth order (zero time delay) cross-correlation can be used as selection criterion for the search of nearest neighbor. Most of the times it is observed that the embedding dimension approximated by maximum in ACF plot, which is an integer value, is not true optimum. Therefore we have upsampled the original series to get optimal fractional embedding dimension. Moreover upsampling generated interpolated points between the data points of the original time series. These data points increased the search space and helped in searching for better nearest neighbor. The details and/or additional experimental results can be found in our journal paper [12] . Here this method is used for the prediction of European symposium on time series prediction (ESTSP) competition's time series.

2 Proposed method for time series prediction

The proposed method consists of following main steps

a. Upsampling the time series
b. Embedding dimension selection
c. Search of best match pattern
d. Multistep-ahead prediction on the base of best match pattern

2.1 Upsampling the time series

Upsampling is a process in digital signal processing, in which data values are interpolated in between the original values of the time series. In statistics it termed as interpolation. The length of the upsampled series becomes the multiple of original length of time series and upsampling order. The upsampling order 1 means no upsampling (the series length is multiplied by 1). Various interpolation methods are being used to upsample the time series e.g. linear interpolation, spline interpolation. We have used the upsampling method proposed by [13]. In this method zeros are inserted in between the original data values of time series then these zeroes are replaced with values suggested by finite impulse response filter (FIR). The design of FIR filter is given in [13]. To illustrate the upsampling a time series of $Sin(80\pi t) + Sin(160\pi t)$ for t = 0 to 2 with step 0.001 is shown in Fig 1. The upsampled time series by 4th order is shown in Fig 2.

2.2 Embedding dimension selection

One of the main issues of nearest neighbor method is the selection of embedding dimension or window size i.e. the number of data values in reference pattern. F. Taken in his theorem [14] says that for sufficiently long time series of scalar observations, one can reconstruct the underlying dynamics by embedding the time series with proper time delays. So there are two parameters of interest, embedding dimension (ED) and time delay (TD). There is no general rule for finding the optimal

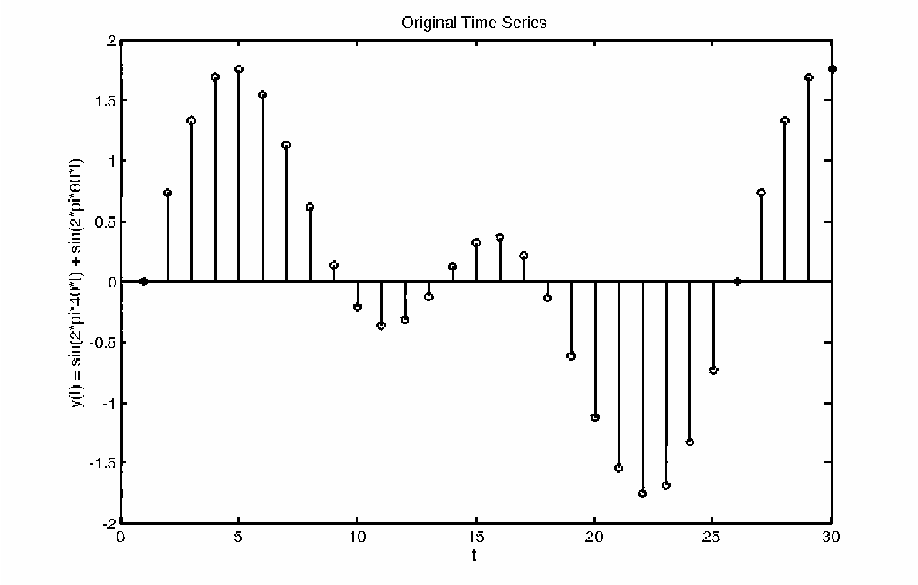

Fig 1: A Sample Time Series

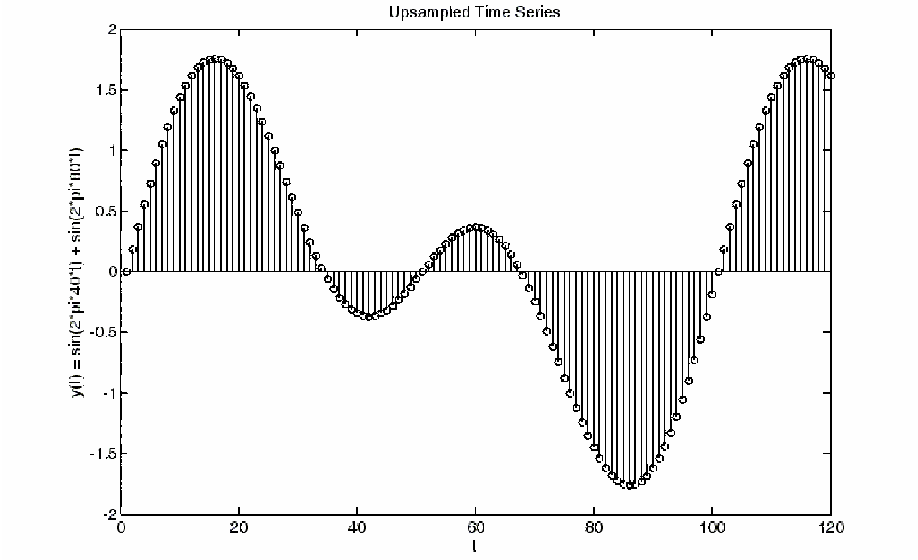

Fig 2: Upsampled Time Series

embedding dimension parameters. Various methods are proposed for calculating TD and ED. Differential entropy based method for determining the optimal embedding parameters of a signal are proposed in [15]. Some existing methods of optimal embedding were reviewed by [16] and they proposed method based on evolution.

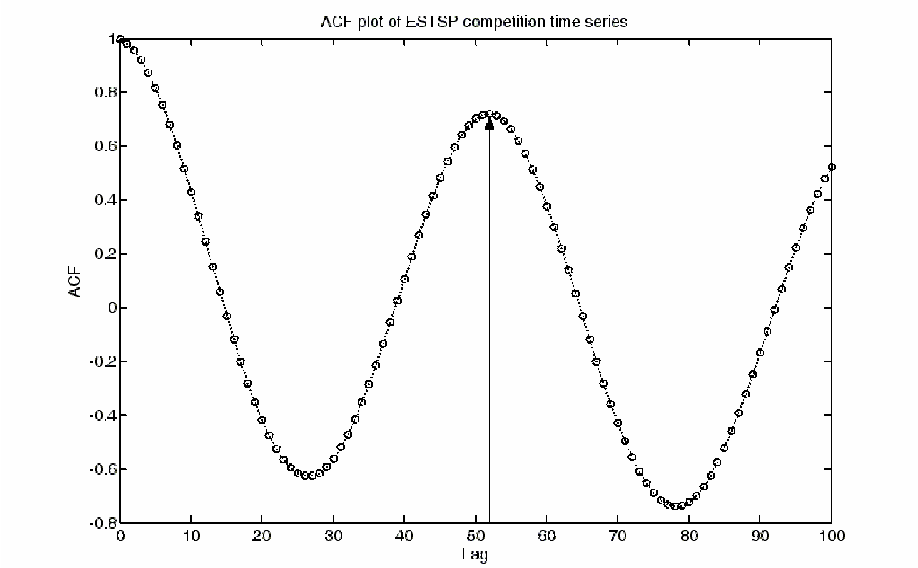

Fig 3: ACF Plot of ESTSP Competition Time Series

M. Small [17] proposed that instead of finding optimal values of individual TD and ED, the product of TD and ED i.e. M = TD × ED can be determined.

We have take TD equal to 1 and found the optimal M (i.e. optimal ED). Periodicity of time series was exploited to find the optimal dimension. Autocorrelation function plot verses time lag is used to get the optimal embedding dimension. The first maximum of the plot occurs at zero time delay. The second maximum gives the best guess of optimal embedding dimension for the time series [18]. The auto-correlation function (ACF) for ESTSP competition series is plotted in Fig 3, which shows the optimal embedding dimension is 52. After upsampling the time series and finding the optimal embedding, the sliding window technique was used. It increased the database of candidate vectors and hence increased the prediction accuracy. The optimal embedding dimension i.e. optimal window size for upsampled ESTSP competition series was also calculated. The table of optimal embedding dimension vs. upsampling order is in result's section (Table 2).

2.3 Search of best match pattern

In nearest neighbor method the most recent data value (reference pattern) of the time series of length equal to optimal embedding dimension is selected and the entire patterns in the past values are matched. We used the sliding window for candidate patterns. For this purpose the original time series is converted into matrix form having rows equal to optimal embedding dimension. The full detail of matrix formation can be seen in [12]. The schematic for nearest neighbor search is given in Fig 4.

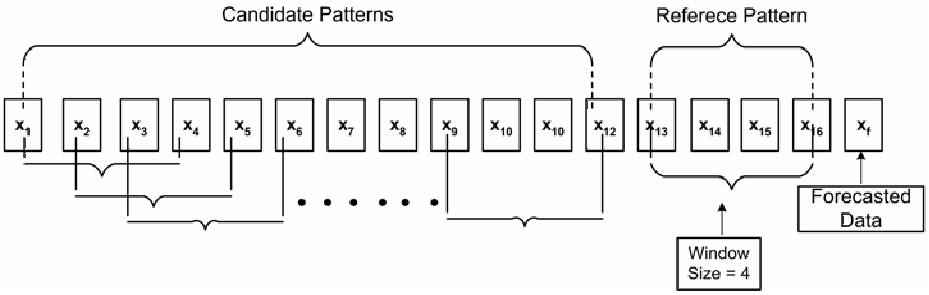

Fig 4: Schematic For Nearest Neighbor Search

The best match pattern is usually selected which has the least Euclidean distance from the reference pattern. The Euclidean distance 'E_d' between two vectors 'X' and 'Y' is given by (1)

$$E_d = \sqrt{\sum_i (X(i) - Y(i))^2} \tag{1}$$

we proposed that instead of using the Euclidean distance as a similarity criterion, normalized zeroth order cross-correlation can be used which is amplitude invariant. If 'X' and 'Y' are two vectors, the normalized cross-correlation with delay 'TD' is defined as

$$Xcorr_{norm}(TD) = \frac{\sum_i X(i)\, Y(i - TD)}{\sqrt{\sum_i X^2(i) * Y^2(i)}} \tag{2}$$

For zeroth order cross-correlation TD is 0. The best match pattern is selected which has maximum normalized cross-correlation value.

2.4 Prediction on the base of best match pattern

The best match pattern is one which has the maximum correlation value, after finding it the prediction of one value is achieved as the next value in the time series of the best matched pattern.

For the multistep-ahead prediction, the reference vector is updated by dropping the oldest value in it and adding the forecasted value at the end so that the length of the reference pattern remains intact. The new reference vector is again matched with candidate patterns and this process is iterated for required prediction steps.

3 Results and Discussion

The proposed method is based on nearest neighbor method. Optimal embedding dimension is estimated using ACF plot and nearest neighbor is selected using zeroth order normalized cross-correlation. The proposed method was evaluated on six benchmark time series, i.e. (1) Santa Fe prediction competition's laser series (2)

annual sunspot number (3) Mackey-Glass (4) Lorenz attractor (5) Henon map (6) electric load of Iowa City. The results can be seen in [12]. The upsampling order 1 to 7 was used. There were two reasons for using the upsampling order till 7, firstly we achieved comparably better results till order 7 and secondly the huge CPU time was required for computation of prediction for further orders. One of our studied series was laser time series introduced in Santa Fe time series prediction competition 1992. In the competition 1000 values was given to predict the next 100 values of laser time series. The best prediction was achieved by Eric wan [19] with normalized mean squared error (NMSE) equal to 0.028 and second best was by T. Sauer [2] with NMSE = 0.080. Our proposed method with upsampling order 7 gives the NMSE = 0.047, which is in between the two top entries of the competition. This shows the usefulness of the method. The effect of upsampling on the prediction error is plotted in Fig 5. The normalized mean squared error without using upsampling and with upsampling of order 7 is shown in Table 1.

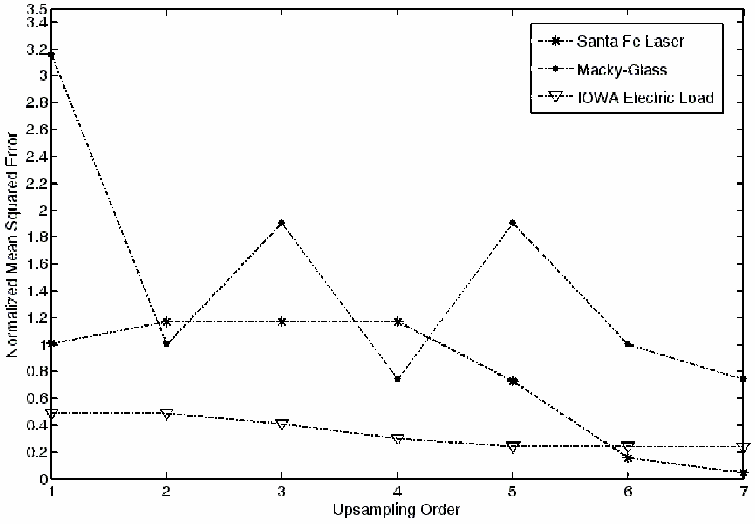

Fig 5: TheEffect of Upsampling on Performance

Table 1: NMSE without and with Upsampling

Time Series	NMSE	
	With out Upsampling	With Upsampling Order 7
Laser	1.0106	0.0468
Sunspot	1.9133	0.8130
Mackey-Glass	3.1561	0.7421
Henon	0.3476	0.9281
Lorenz	1.3707	0.9425
IOWA Electric	0.4915	0.2347

As described earlier the optimal embedding dimension for ESTSP competition series is found as 52. The optimal embedding dimension changes, when the series is upsampled and it is not the integer multiple of 52 (optimal dimension for original time series). The optimal dimension with upsampling order is shown in Table 2. The effective dimension (optimal embedding dimension of upsampled series divided by upsampling order) is shown in parenthesis. The effective optimal dimension is fractionally close to 52. We couldn't get these fractional dimensions without upsampling.

For ESTSP competition series prediction, first we evaluated our proposed method to forecast the values from 801 to 875 (taken as a test set) using initial 800 values of ESTSP competition series. The mean squared error found using upsampling order 1 to 7 for test set is shown in Table 3; which shows that MSE reduces when upsampling order is more than 1. There was no more reduction in MSE after upsampling order 2. The mean squared error equal to 0.013 has been achieved for multistep-ahead prediction of values for steps 801 to 875. The prediction plot of test set is illustrated as Fig 6.

Table 2: The Optimal Dimension for ESTSP Time Series

Sr. No.	Upsampling order	Optimal Embedding (Effective dimension in Parenthesis)
1	1	052 (52.00)
2	2	104 (52.00)
3	3	155 (50.66)
4	4	207 (51.55)
5	5	259 (51.80)
6	6	311 (51.83)
7	7	362 (51.71)

Table 3: MSE in Prediction of Test Set of ESTSP Time Series

Sr. No.	Upsampling order	MSE
1	1	0.018
2	2	0.013
3	3	0.013
4	4	0.013
5	5	0.013
6	6	0.013
7	7	0.013

For submission of competition entry we used upsampling order 7 in our proposed method. Although the MSE for test set did not decrease after upsampling order 2, the reasons for choosing upsampling order 7 is that in above mentioned experiments

order 7 was giving good prediction. However in ESTSP competition's series case the values of the prediction did not change in first decimal place whether order 2 or 7 was used. The predicted 100 values of ESTSP competition time series and plot is shown in Fig 7. As required in the competition the 50 predicted values are illustrated in Fig 8.

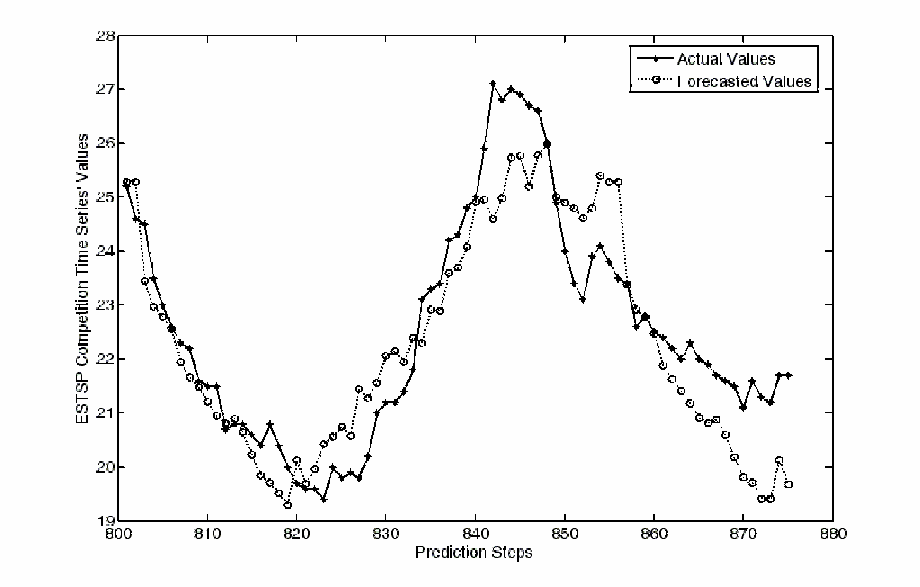

Fig 6: ESTSP Series Prediction from 801 to 875 as a Test Set

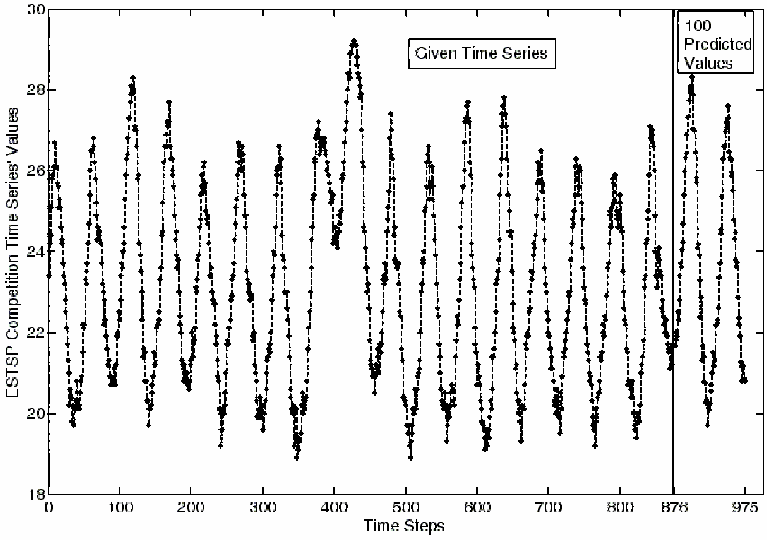

Fig 7: Prediction of 100 values of ESTSP Competition Time Series

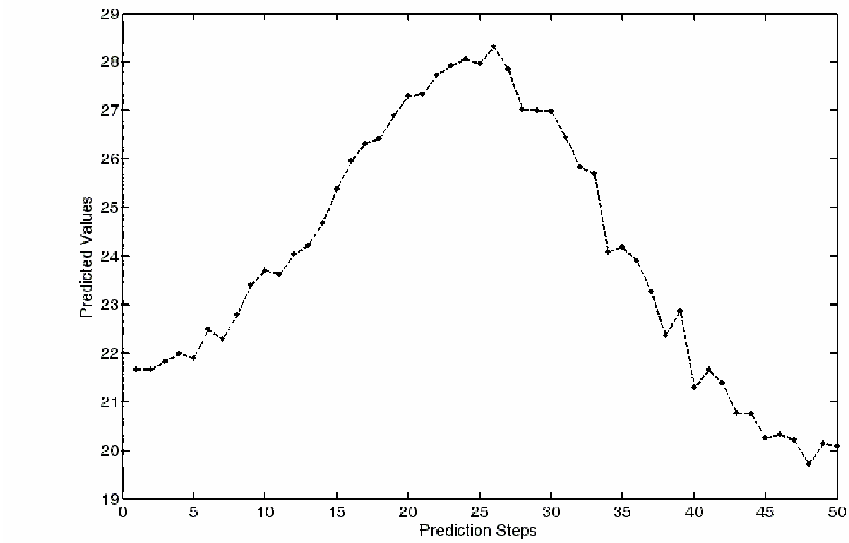

Fig 8: Close Look of ESTSP Competition 50 Predicted Values

Conclusion

In this paper nearest neighbor based method, which is based on upsampling of time series and cross-correlation based similarity measure is used to predict the ESTSP competition time series. The MSE for test set of ESTSP time series, i.e. for prediction of values from time step 801 to 875 is 0.0131. Proposed method has been used to submit the competition time series prediction.

References

[1] T. Cover and P. Hart, Nearest neighbor pattern classification, *IEEE Transactions on Information Theory*, 13(1), 21-27, IEEE, 1967.

[2] T. Sauer, Time series prediction by using delay coordinate embedding, in *Time series prediction: Forecasting the future understanding the past*, A. S. Weigend and N. A. Gershenfeld, editors: Addison Wesley, 1993.

[3] N. Meade, A comparison of the accuracy of short term foreign exchange forecasting methods, *International Journal of Forecasting*, 18(67-83, Elsevier, 2002.

[4] F. Fernandez-Rodriguez, S. Sosvilla-Rivero, and J. Andrada-Felix, Exchange-rate forecasts with simultaneous nearest-neighbour methods: Evidence from the ems, *International Journal of Forecasting*, 15(383-392, Elsevier, 1999.

[5] F. J. Mulhern and R. J. Carrara, A nearest neighbor model for forecasting market response, *International Journal of Forecasting*, 10(191-207, Elsevier, 1994.

[6] T. Raicharoen and C. Lursinsap, A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (poc-nn) algorithm, *Pattern Recognition Letters*, 26(1554-1567, Elsevier, 2005.

[7] C. Domeniconi, J. Peng, and D. Gunopulos, Locally adaptive metric nearest-neighbor classification, *IEEE transaction on pattern analysis and machine intelligence*, 24(9), IEEE, 2002.

[8] B. L. Smith, Effect of parameter selection on forecast accuracy and execution time in nonparametric regression, in proceedings of *IEEE intelligent Transportation system Conference*, USA, 2000.

[9] S. B. Day, Combining nearest neighbor classifiers through multiple feature subsets, in proceedings of *5th International Conference on Machine Learning*, Madison WI, 1998.

[10] A. Djouadi and E. Bouktache, A fast algorithm for nearest-neighbor classifier, *IEEE transaction on pattern analysis and machine intelligence*, 19(3), IEEE, 1997.

[11] T. Hastie and R. Tibshirani, Discriminant adaptive nearest neighbor classification, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 18(6), IEEE, 1996.

[12] S. R. Abbas and M. Arif, Long range time series forecasting by upsampling and using cross-correlation based selection of nearest neighbor, *International Journal of Pattern Recognition and Artificial Intelligence*, 20(8), 1261-1278, World Scientific, 2006.

[13] G. Oetken, T. W. Parks, and H. W. Schussler, New results in the design of digital interpolators, *IEEE transaction on Acoustics, Speech and Signal Processing*, 23(301-309, IEEE, 1975.

[14] F. Takens, Detecting attractors in turbulence, in *Lecture notes in mathematics*, vol. 898, D. Rand and L. S. Young, editors. Berlin: Springer-Verlag, 1981.

[15] T. Gautama, D. P. Mandic, and M. M. Van Hulle, A differential entropy based method for determining the optimal embedding parameters of a signal, in proceedings of *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2003.

[16] V. Babovic, M. Keijzer, and M. Stefansson, Optimal embedding using evolutionary algorithms, in proceedings of *4th International Conference on Hydroinformatics*, Iowa City, 2000.

[17] M. Small, Optimal embedding parameters: A modeling paradigm, *Physica D*, 194(283-296, Elsevier, 2004.

[18] R. Abbas, W. Aziz, and M. Arif, Modified nearest neighbour algorithm for time series forecasting, *Sci. Int (Lahore), Pakistan*, 17(3), 191-194, 2005.

[19] E. Wan, Time series prediction using a neural network with embedded tapped delay-lines, in *Time series prediction: Forecasting the future understanding the past*, A. S. Weigend and N. A. Gershenfeld, editors: Addison Wesley, 1993.

Variable Scaling for Time Series Prediction

Francesco Corona and Amaury Lendasse *

Helsinki University of Technology - Laboratory of Computer and Information Science
P.O. Box 5400, 02015 HUT - Finland

Abstract. In this paper, variable selection and variable scaling are used in order to select the best regressor for the problem of time series prediction. Direct prediction methodology is used instead of the classic recursive methodology. Least Squares Support Vector Machines (LS-SVM) are used in order to avoid local minimal in the training phase of the model. The global methodology is applied to the time series competition dataset.

1 Introduction

Time series forecasting is a challenge in many fields. In finance, experts forecast stock exchange courses or stock market indices; data processing specialists forecast the flow of information on their networks; producers of electricity forecast the load of the following day. The common point to their problems is the following: how can one analyse and use the past to predict the future?

Many techniques exist for the approximation of the underlying process of a time series: linear methods such as ARX, ARMA, etc. [1], and nonlinear ones such as artificial neural networks [2]. In general, these methods try to build a model of the process. The model is then used on the last values of the series to predict the future values. The common difficulty to all the methods is the determination of sufficient and necessary information for an accurate prediction.

A new challenge in the field of time series prediction is the Long-Term Prediction: several steps ahead have to be predicted. Long-Term Prediction has to face growing uncertainties arising from various sources, for instance, accumulation of errors and the lack of information [2].

In this paper, a global methodology to perform direct prediction is presented. It includes variable selection and variable scaling. The variable selection criterion is based on a Nonparametric Noise Estimation (NNE) performed by Delta Test.

In this paper, Least Squares Support Vector Machines (LS-SVM) are used as nonlinear models in order to avoid local minima problems [3].

Section 2 presents the prediction strategy for the Long-Term Prediction of Time Series. In Section 3 Delta Test is introduced. Section 4 introduces the variable selection and scaling selection. The prediction model LS-SVM is briefly summarized in Section 5 and experimental results are shown in Section 6 using the competition dataset.

*Part the work of F. Corona and A. Lendasse is supported by the project of New Information Processing Principles, 44886, of the Academy of Finland. The work of A. Lendasse is supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

2 Time Series Prediction

The time series prediction problem is the prediction of future values based on the previous values and the current value of the time series (see Equation 1). The previous values and the current value of the time series are used as inputs for the prediction model. One-step ahead prediction is needed in general and is referred to as Short-Term Prediction. But when multi-step ahead predictions are needed, it is called a Long-Term Prediction problem.

Unlike the Short-Term time series prediction, the Long-Term Prediction is typically faced with growing uncertainties arising from various sources. For instance, the accumulation of errors and the lack of information make the prediction more difficult. In Long-Term Prediction, performing multiple step ahead prediction, there are several alternatives to build models. In the following sections, two variants of prediction strategies are introduced and compared: the Direct and the Recursive Prediction Strategies.

2.1 Recursive Prediction Strategy

To predict several steps ahead values of a time series, Recursive Strategy seems to be the most intuitive and simple method. It uses the predicted values as known data to predict the next ones. In more detail, the model can be constructed by first making one-step ahead prediction:

$$\hat{y}_{t+1} = f_1(y_t, y_{t-1}, ..., y_{t-M+1}), \tag{1}$$

where M denotes the number inputs. The regressor of the model is defined as the vector of inputs: $y_t, y_{t-1}, ..., y_{t-M+1}$. It is possible to use also exogenous variables as inputs in the regressor, but they are not considered here in order to simplify the notation. Nevertheless, the presented global methodology can also be used with exogenous variables.

To predict the next value, the same model is used:

$$\hat{y}_{t+2} = f_1(\hat{y}_{t+1}, y_t, y_{t-1}, ..., y_{t-M+2}). \tag{2}$$

In Equation 2, the predicted value of $\hat{y}_{t+1}$ is used instead of the true value, which is unknown. Then, for the H-steps ahead prediction, $\hat{y}_{t+2}$ to $\hat{y}_{t+H}$ are predicted iteratively. So, when the regressor length M is larger than H, there are $M - H$ real data in the regressor to predict the H^{th} step. But when H exceeds M, all the inputs are the predicted values. The use of the predicted values as inputs deteriorates the accuracy of the prediction.

2.2 Direct Prediction Strategy

Another strategy for the Long-Term Prediction is the Direct Strategy. For the H-steps ahead prediction, the model is

$$\hat{y}_{t+h} = f_h(y_t, y_{t-1}, ..., y_{t-M+1}) \text{ with } 1 \leq h \leq H. \tag{3}$$

This strategy estimates H direct models between the regressor (which does not contain any predicted values) and the H outputs. The errors in the predicted values are not accumulated in the next prediction. When all the values, from $\hat{y}_{t+1}$ to $\hat{y}_{t+H}$, need to be predicted, H different models must be built. The direct strategy increases the complexity of the prediction, but more accurate results are achieved.

3 Nonparametric Noise Estimator using the Delta Test

Delta Test (DT) is a technique for estimating the variance of the noise, or the mean square error (MSE), that can be achieved without overfitting [4]. The evaluation of the NNE is done using the DT estimation introduced by Stefansson in [5].

Given N input-output pairs: $(x_i, y_i) \in \mathbb{R}^M \times \mathbb{R}$, the relationship between x_i and y_i can be expressed as:

$$y_i = f(x_i) + r_i, \tag{4}$$

where f is the unknown function and r is the noise. The Delta Test estimates the variance of the noise r.

The DT is useful for evaluating the nonlinear correlation between two random variables, namely, input and output pairs. The DT has been introduced for model selection but also for variable selection: the set of inputs that minimizes the DT is the one that is selected. Indeed, according to the GT, the selected set of variables is the one that represents the relationship between variables and output in the most deterministic way.

DT is based on hypotheses coming from the continuity of the regression function. If two points x and x' are close in the input space, the continuity of regression function implies the outputs $f(x)$ and $f(x')$ will be close enough in the output space. Alternatively, if the corresponding output values are not close in the output space, this is due to the influence of the noise.

Let us denote the first nearest neighbor of the point x_i in the set $\{x_1, \ldots, x_N\}$ by x_{NN}. Then the delta test, δ is defined as:

$$\delta \quad = \quad \frac{1}{2N} \sum_{i=1}^{N} \left| y_{NN(i)} - y_i \right|^2, \tag{5}$$

where $y_{NN(i)}$ is the output of $x_{NN(i)}$. For the proof of the convergence of the Delta Test, see [4].

4 Variable and Scaling Selection

Variable scaling is a usual preprocessing step in both function approximation and time series analysis. In scaling, weights are used to reflect the relevance of the input variables to the output to be estimated. That is, scaling seeks

for redundant inputs and assigns them low weights to reduce the corresponding influence on the learning process. In such a context, it is clear that variable selection is a particular case of scaling: by weighting irrelevant variables by zero we are, indeed, enforcing selection. For the sake of brevity, only the main concepts referring to the regression problem are presented here. Nevertheless, the extension to time series analysis is trivial.

4.1 Scaling the Input Space with Mahalanobis Matrices

The Mahalanobis distance $d_M(x_i, x_j)$ of two d-dimensional observations x_i, x_j is a proximity (or 'similarity') measure defined on the dependencies between the embedding dimensions. Formally, $d_M(x_i, x_j)$ extends the traditional Euclidean distance $d(x_i, x_j) = [(x_i - x_j)^T(x_i - x_j)]^{1/2}$ transforming the observations subspace by means of a $(d \times d)$ full-rank matrix M:

$$d(x_i, x_j) = [(x_i - x_j)^T M(x_i - x_j)]^{1/2}, \tag{6}$$

From the previous equation, it is evident that: i) if $M = I$ then the original Euclidean metric is retained, and ii) if M is a $(d \times d)$ diagonal matrix then the original space is simply rescaled according to the diagonal elements. Matrix M is also symmetric and semi-definite positive, by definition. Moreover, the Mahalanobis matrix M can be factorized as:

$$M = S^T S, \tag{7}$$

with a matrix S that can linearly map the observations into the subspace spanned by the eigenvectors of the transformation. The learned metric in the projection subspace is still the Euclidean distance, that is:

$$d(x_i, x_j) = [(x_i - x_j)^T M(x_i - x_j)]^{1/2} = [(Sx_i - Sx_j)^T(Sx_i - Sx_j)]^{1/2}, \tag{8}$$

where, by restricting S to be a non-square ($s * d$, with $s < d$) matrix, the transformation performs both a reduction of the dimensionality and the scaling of the original input subspace. The resulting subspace has an induced global metric of lower rank suitable for reducing the 'curse of dimensionality'.

In this paper, we use a diagonal matrix M that is optimized in order to minimize the delta test estimation in the scaled space define by S. Details about the optimization method are given the the experiments section.

5 Nonlinear Models

In this paper, Least Squares Support Vector Machines (LS-SVM) are used as nonlinear models [3], which are defined in their primal weight space by [6, 7]

$$\hat{y} = \omega^T \varphi(\mathbf{x}) + b, \tag{9}$$

where $\varphi(\mathbf{x})$ is a function, which maps the input space into a higher-dimensional feature space, $\mathbf{x}$ is the vector of inputs. ω and b are the parameters of the model. The optimization problem can be formulated as

$$\min_{\omega,b,e} J(\omega, e) = \frac{1}{2}\omega^T\omega + \gamma\frac{1}{2}\sum_{i=1}^{N} e_i^2, \tag{10}$$

$$\text{subject to} \quad y_i = \omega^T\varphi(\mathbf{x}_i) + b + e_i, i = 1, ..., N, \tag{11}$$

and the solution is

$$h(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b. \tag{12}$$

In the above equations, i refers to the index of a sample and $K(\mathbf{x}, \mathbf{x}_i)$ is the kernel function defined as the dot product between the $\varphi(\mathbf{x})^T$ and $\varphi(\mathbf{x})$. Training methods for the estimation of the ω and b parameters can be found in [6].

6 Experimental Results

In this paper, the ESTSP2007 competition dataset is used as an example. It includes a total of 875 values. The dataset is shown in Figure 1.

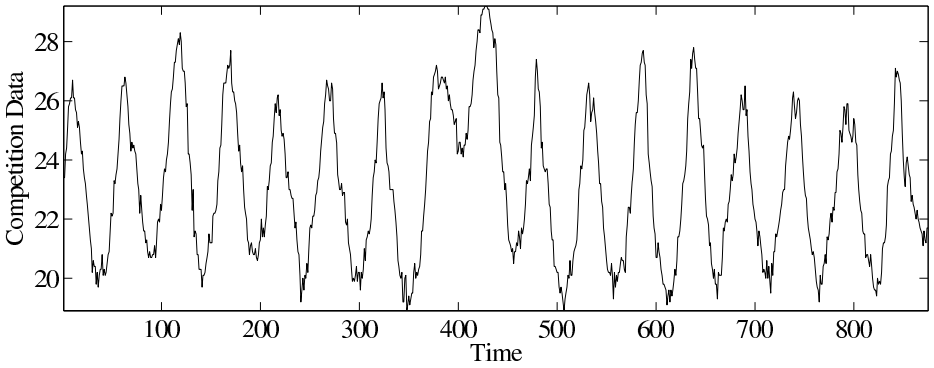

Fig. 1: Competition dataset.

In order to test the methodology, the dataset is divided into two sets, a small learning set and the global learning set. The small learning set consists of 465 first values and the global learning set consists in the 875 values. The regressor size is set to 10 after many trial and error experiments. The small learning set is used in order to evaluate the performances of the methodology.

The variable scaling is selected in order to minimize the Delta Test estimation. Because the DT is not continuous with respect to the scaling factors, a

forward-backward optimization is used. The variable scaling coefficients are selected between a set of discrete values: [0 0.1 0.2 ... 0.9 1]. This discretization provides satisfactory results and reduces computational time.

The variable scaling is performed for each of the 50 prediction models from equation 3 used in direct prediction methodology. The estimation of the NNE (using Delta Test) are shown in Figure 2.

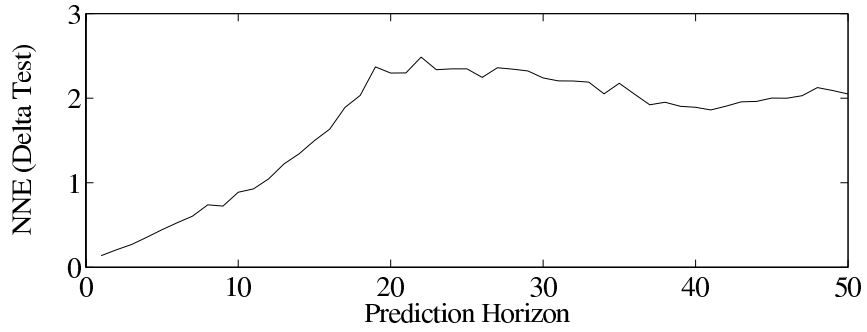

Fig. 2: Estimation of the NNE (using Delta Test) with respect to the horizon of prediction.

The result of the 50 step-ahead prediction is represented in figure 3.

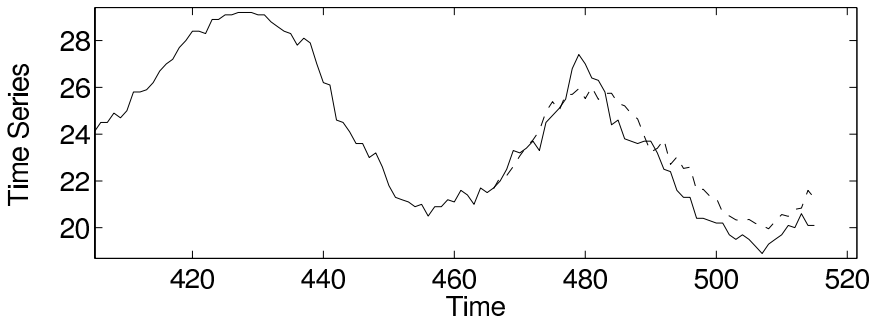

Fig. 3: Comparison between the time series (solid line) and the prediction (dashed line)

Then, the same methodology is used with the global learning set in order to predict the competition values. The estimation of the NNE (using Delta Test) are shown in Figure 4.

The result of the 50 step-ahead prediction is represented in figure 5.

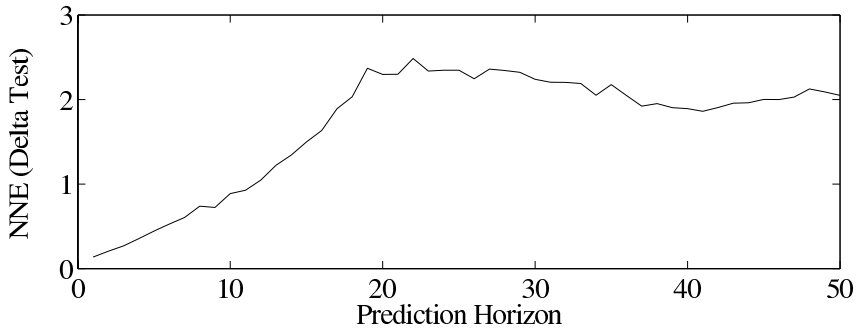

Fig. 4: Estimation of the NNE (using Delta Test) with respect to the horizon of prediction.

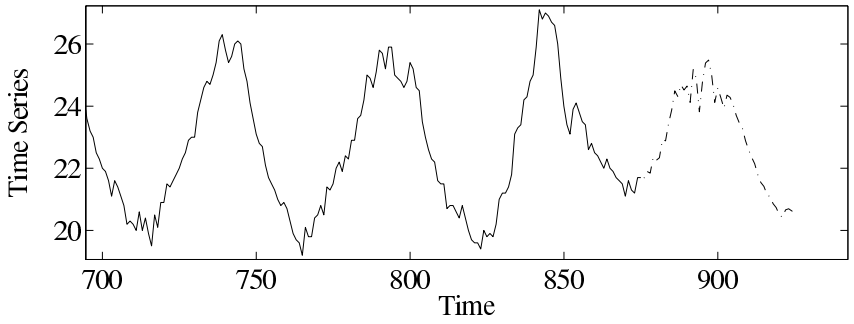

Fig. 5: Prediction of 50 next values of the competition dataset. The real values are presented by the solid line and the dashed one presents the prediction.

7 Conclusion

In this paper, we have presented a methodology for the longterm prediction of time series.

This methodology uses direct prediction methodology. This increases the computational time but improves the quality of the results.

In order to perform the variable scaling, Delta Test estimation is used. The scaling that minimized the NNE is selected. To reduce the computational time, a discrete scaling is used and a forward-backward optimization is selected.

Further research will be done to improve the minimization of the NNE estimation. Other experiments will be performed in the fields of time series prediction and function approximation.

References

[1] L. Ljung. *System identification theory for User*. Prentice-Hall, Englewood CliPs, NJ, 1987.

[2] A.S. Weigend and N.A. Gershenfeld. *Times Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley Publishing Company, 1994.

[3] Johan A K Suykens, Tony Van Gestel, Jos De Brabanter, Bart De Moor, and Joos Vandewalle. *Least Squares Support Vector Machines*. World Scientific Publishing Co., Pte, Ltd. (Singapore), November 2002.

[4] A. J. Jones. New tools in non-linear modeling and prediction. *Computational Management Science*, 1:109–149, 2004.

[5] Adalbjörn Stefansson, N. Koncar, and Antonia J. Jones. A note on the gamma test. *Neural Computing & Applications*, (5(3)):131–133, 1997.

[6] Available from http://www.esat.kuleuven.ac.be/sista/lssvmlab/.

[7] Johan A. K. Suykens, Jos De Brabanter, L. Lukas, and Joos Vandewalle. Weighted least squares support vector machines: robustness and sparse approximation. *Neurocomputing*, 48:85–105, 2002.

Comparison of FDA based Time Series Prediction Methods

Tuomas Kärnä and Amaury Lendasse

Helsinki University of Technology - Neural Networks Research Center
P.O. Box 5400 FI-02015 - FINLAND

Abstract. Functional Data Analysis (FDA) provides an important addition to traditional data analysis methods. In FDA a function is fitted to the data and the fitting coefficients are examined instead of the original data. The function fitting, however, is not a straight forward task. The choice of the function space is often crucial and the fitting may involve unknown parameters that need to be determined. This paper presents a comparison of different FDA based methods for time series prediction. The experimented function types are B-splines, Wavelets and Gaussian kernels. In all cases k Nearest Neighbor (k-NN) model is used for the prediction. Furthermore, some input selection methods are experimented to improve the k-NN performance.

1 Introduction

Common time series prediction methods operate in time space and estimate future values directly. A rather different approach is to utilize Functional Data Analysis (FDA) [5] in this task. The idea is to fit some function to the data points and work with the fitting coefficients instead of the original data. In this case, however, one needs to find a suitable set of functions. Smooth functions might be a good guess, if the data is smooth or if the data is very noisy. In the latter case the fitting might result in some noise cancellation. However, it is very difficult to know a priori which basis is suitable for certain data. Furthermore, the fitting usually involves some unknown parameters that need to be determined.

In this paper, we present a comparison of three function types, B-splines, Wavelets and Gaussian kernels in FDA time series prediction task. For all of the functions a k-Nearest Neighbor [3] (k-NN) prediction model is trained. k-NN is suitable for this task because of its robustness and small computational load. However, the drawback of k-NN is that it is sensitive to scaling of the inputs. For this purpose some input selection methods are also examined [8].

The proposed FDA time series method is described in Section 2. Section 3 briefly presents the function types and some of their properties along with a few notions related to FDA applications. The experiments are outlined in Section 4 followed by results and discussion in Section 5. Finally, Section 6 concludes the paper.

2 FDA for Time Series Prediction

When applying FDA to time series prediction, the basic idea is to cast the original prediction problem from time space to some function space. In practice this

means that we work with the function coefficients instead of original data points. The functional representation has some advantages in comparison to traditional prediction schemes, such as dimensionality reduction and the possibility to work with irregularly sampled data.

Consider a set of observations $(x_i, y_i)_{i=1}^n$ in a closed interval $x_i \in [a, b]$. First, the data is divided into input windows I_h and output windows O_h of equal length L. Output is the strictly following window of the corresponding input. The sets are defined as,

$$\begin{aligned} I_h := (x_i^h, y_i^h)_{i=1}^{m_h} &= \{(x_i, y_i) \,|\, a + (h-1)\delta \quad \leq x_i < a + (h-1)\delta + L\} \\ O_h := (\hat{x}_i^h, \hat{y}_i^h)_{i=1}^{\hat{m}_h} &= \{(x_i, y_i) \,|\, a + (h-1)\delta + L \leq x_i < a + (h-1)\delta + 2L\}, \end{aligned}$$

where $h = 1, \ldots, N$ and δ stands for the shift between two sequential windows. The number of windows is $N = \lfloor (b - a - 2L)/\delta \rfloor + 1$. The x_i^h values are shifted so that they all belong to an interval $[0, L)$ for all h. Thus, the sample sets are located in the same interval in the time space

If the data is regularly sampled, the sampling time is a natural choice for δ. In this case the sets I_h and O_h intersect, given by $I_{h+L/\delta+1} = O_h$ for $h = 1, \ldots, (b - a - 3l)/\delta$. Regular sampling also implies that there is exactly L data pairs in each set, i.e. $m_h = \hat{m}_h = L$. Furthermore, the x_i coincide and we can write $x_i^h = x_i$ for all h. For simplicity, it is assumed from now on that the data is regularly sampled.

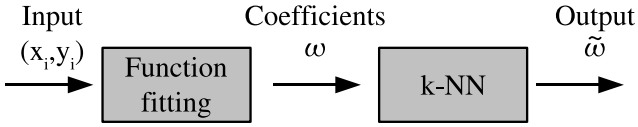

Fig. 1: Prediction method. A function is fitted to the data points and the prediction is done in the function space using k-NN. The output is also a set of coefficients.

2.1 Function Fitting

The outline of the prediction method is presented in Figure 1. An interpolating function is fitted for each window I_h and O_h. To be more specific, it is assumed that there exits some regular function $f \in L^2$ so that $y_i = f(x_i) + s_i$, where s_i stands for observation noise. Working with L^2 in practice, however, is impossible because it is infinite dimensional. For this reason it is necessary to take some finite dimensional subspace $\mathcal{A} \subset L^2$ instead. Knowing the truncated basis φ_i of $\mathcal{A}$ we can approximate f by minimizing the mean square fitting error,

$$\min J(\mathbf{w}) \quad = \quad \sum_{i=1}^{m} (y_i - \hat{f}(x_i))^2 \text{ with } \hat{f}(x) = \sum_{l=1}^{q} w_l \varphi_l(x) \tag{1}$$

where w_l stands for the fitting coefficients and q is the dimension of $\mathcal{A}$. The coefficients w_l define the approximation $\hat{f}$ uniquely and can thus be used as "new" data in further analysis.

When working with the function coefficients w_l, orthogonality of the basis must be taken into consideration. One good property of functional spaces is that the Euclidian operations (such as norm or innerproduct) can also be extended to the functions. Therefore it is natural to require that the innerproduct of the coefficients (i.e. $\langle \mathbf{w}, \mathbf{v} \rangle = \mathbf{w}^T\mathbf{v}$) is equal to the innerproduct of the functions ($\int_{\mathbb{R}} f(x)g(x)dx$). A direct substitution yields,

$$\int_{\mathbb{R}} f(x)g(x)dx = \int_{\mathbb{R}} \sum_{l=1}^{q} w_l\varphi_l(x) \sum_{k=1}^{q} v_k\varphi_k(x)dx = \mathbf{w}^T\Phi\mathbf{v}$$

where $\Phi_{i,j} = \int_{\mathbb{R}} \varphi_i(x)\varphi_j(x)dx$. This implies that if the basis is orthonormal, the matrix Φ becomes an identity matrix and the requirement is automatically met. Otherwise, a linear transformation $\boldsymbol{\omega} = \mathbf{U}\mathbf{w}$ must be applied. Here the matrix $\mathbf{U}$ is a Cholesky decomposition of $\boldsymbol{\Phi} = \mathbf{U}^T\mathbf{U}$ and we get $\boldsymbol{\omega}^T\boldsymbol{\nu} = (\mathbf{U}\mathbf{w})^T\mathbf{U}\mathbf{v} = \mathbf{w}^T\mathbf{U}^T\mathbf{U}\mathbf{v}$. The obtained input and output coefficient sets are denoted as $\mathcal{I}_h = \boldsymbol{\omega}(I_h)$ and $\mathcal{O}_h = \boldsymbol{\omega}(O_h)$, respectively.

2.2 k-NN prediction

k-NN clustering is a widely used for example in pattern recognition [3] and time series prediction [8]. Here, k-NN prediction for functional data is presented. For working with original data directly, see [8].

Our goal is to build a prediction model $P : \mathcal{I}_h \mapsto \mathcal{O}_h$. With k-NN, the estimate for the future is obtained as a mean of the k most similar outputs. I.e. given the training pairs $(\boldsymbol{\omega}_i, \boldsymbol{\nu}_i)$, $\boldsymbol{\omega}_i \in \mathcal{I}_h$, $\boldsymbol{\nu}_i \in \mathcal{O}_h$ and a previously unknown input $\boldsymbol{\omega}$ we get the estimate,

$$\tilde{\boldsymbol{\nu}} \quad = \quad P(\boldsymbol{\omega}) = \frac{1}{k} \sum_{\{\boldsymbol{\nu}_i | \boldsymbol{\omega}_i \in \mathcal{N}_k(\boldsymbol{\omega})\}} \boldsymbol{\nu}_i,$$

where the set $\mathcal{N}_k(\boldsymbol{\omega}) \subset \mathcal{I}_h$ is the k nearest neighbors of $\boldsymbol{\omega}$. Usually the Euclidian metric is used to compute the distance between samples. The number of neighbors, k, is unknown and must be validated separately. Quality of the prediction is measured by evaluating the predicted function at the data locations and taking mean square of the errors.

2.2.1 Input Selection

Due to its simplicity, k-NN is computationally very cheap which makes it suitable for handling large data sets and time consuming parameter optimization tasks. However, it is well known that the performance of k-NN can be greatly decreased if the data is noisy or if there exists a lot of irrelevant information.

Because k-NN is based on pair wise distances, it may be useful to normalize the inputs to zero mean and unit variance. This ensures that each dimension

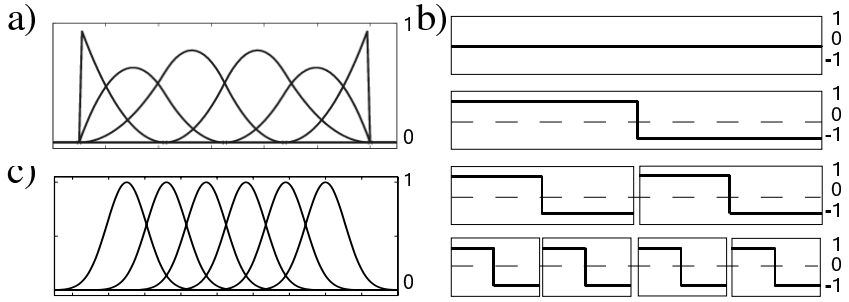

Fig. 2: Basis functions. a) B-Splines b) Haar Wavelets for 8 dimensional data c) Gaussian kernels

will contribute equally to the distances. Normalization is helpful in situations were certain dimensions have large variance and thus dominate the choice of neighbors. However, it does not resolve the problem of irrelevant information.

As another choice, input selection [8] or scaling [9] can be used. In the case of input selection the pair wise distances $|\boldsymbol{\omega} - \boldsymbol{\omega_i}|$ are calculated using only a subset of the data dimensions. In scaling, on the other hand, each dimension is scaled separately before the computation of the distances. The latter can be formulated as $[\omega_1, \omega_2, \ldots, \omega_q] \hookleftarrow [\alpha_1\omega_1, \alpha_2\omega_2, \ldots, \alpha_q\omega_q], \quad \alpha_i \in [0, 1]$. Thus input selection is actually a special case of the scaling where the scaling parameters α_i are restricted to be either 1 or 0.

Input selection increases the computational load significantly. For example, running an exhaustive search, i.e. trying out all the possible combinations of $\alpha_i \in \{0, 1\}$, increases the work load by a factor $2^q - 1$, which quickly leads into unthinkable running times as q grows. For scaling, unfortunately, the situation is not any better, because the optimization problem is very difficult.

3 Functional Spaces

3.1 B-splines

Splines are piecewise polynomials developed for data interpolation. This section presents a short revision of splines in the B-form. B-splines provide a convenient basis function representation to the piecewise polynomials. For more detailed information and discussion on algorithms see [2].

The B-spline basis $B_{i,d}$ is uniquely defined by a order d and a non-decreasing knots sequence t_i with possible multiplicities. The basis can be computed recursively [2],

$$
\begin{aligned}
B_{i,1}(x) &= \begin{cases} 1 & \text{if } t_i \le x < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \\
B_{i,d}(x) &= \frac{x - t_i}{t_{i+d-1} - t_i} B_{i,d-1}(x) + \frac{t_{i+d} - x}{t_{i+d} - t_{i+1}} B_{i,d-1}(x).
\end{aligned}
$$

The basis has small support, since $b_{i,d}(x) = 0$ for all $x \notin [t_i, t_{i+d}]$. The order d is related to the smoothness constraints of the spline. The larger the d, the smoother the fitting. In fact, if a knot has multiplicity r, i.e. $t_j = t_{j+1} = \ldots = t_{j+r-1}$, then the number of continuous constraints at t_j equals to $d - r$. So, if $r = d$, the spline becomes discontinuous.

It is convenient to set the spline to be discontinuous at the first and last data location and as smooth as possible in between. Thus the first and the last knot has multiplicity d and the rest are simple. In this case, we have a handy relation,

$$number\ of\ knots = q + d$$

where q is the number of basis functions. The location of the remaining knots are set so that the fitting becomes as accurate as possible. There are several methods available, the one used in this paper is presented in [7]. An example of 3rd order B-spline basis is presented in Figure 2.

In FDA the B-splines have some clear advantages; The fitting is guaranteed to be smooth by definition and it is well suited for reducing data dimensionality. However, the two unknown parameters q and d need to be to optimized. Furthermore, the basis is not orthonormal, so the Cholesky decomposition described in Section 2.1 must be applied.

3.2 Wavelets

Wavelet transformation resembles Fourier transform in that both provide a time-frequency description of the data [1]. Wavelets, however, are able to encode spatial information as well, which make them appealing for multiresolutional signal processing. Given some mother wavelet function ψ, the basis can be written as,

$$\psi_{i,j}(x) = 2^{-i/2}\psi(2^{-i}x - j),$$

where i and j are integers that represent variation in frequency and spatial location, respectively. The wavelet basis functions form an orthonormal basis for L^2. An example of the simplest wavelet, Haar (or Daubechies 1) is presented in Figure 2. There are many wavelet families available with different properties, such as Daubechies (in [1] these are called compactly supported wavelets), Biorthogonal and Mayer wavelets.

Usually there is no need to compute the transformations using the basis functions. Instead the transformation can be obtained efficiently using digital sub band filtering [1]. In this case, however, we have to assume that the data has a constant sampling rate. Furthermore, due to the nature of the basis, data length should be a power of 2. Otherwise some data points must be generated during the filtering process, and the transformation actually becomes longer than the original data. Thus discrete wavelet decomposition cannot reduce dimensionality.

Although dimensionality cannot be reduced, the wavelets have some good properties for FDA. First of all, orthonormality of the basis is always desirable, and on the other hand there are no additional parameters involved. Thus only the window length and wavelet type must be tested.

3.3 Gaussian kernels

The Gaussian fitting is the simplest function type presented here. The basis functions are Gaussian kernels,

$$\varphi_i(x) = e^{\frac{\|x-t_i\|^2}{2\sigma_i^2}}, \qquad (2)$$

centered at t_i and with width parameter σ_i, $i = 1, \ldots, q$. Thus there are two unknown parameters for each kernel. In total we end up with $2 + 2q$ free parameters, (window length, number of kernels, centers and widths) which is considerably more than with Wavelets or B-splines.

One way to ease the parameter optimization is to fix the locations t_i so that they are equally distributed on the interval of the data. This is justified in the case of regularly sampled data, although it may be far away from the optimal. Furthermore, all the kernels can be share the same width parameter σ. This reduces the number of unknown parameters down to 3 which is feasible for a grid search, for example. The grid search optimization was presented in [10]. An example of such Gaussian functions is presented in Figure 2.

When the locations and widths are known, the fitting weights are obtained by solving the problem (1). The solution is the well-known pseudo inverse $\mathbf{w} = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{y}$ [6], where $\mathbf{y} = [y_1, y_2, \ldots, y_m]^T$ are the values to be fitted and $\mathbf{G}_{i,j} = \varphi_j(x_i)$. In practice the matrix $\mathbf{G}^T\mathbf{G}$ tends to become singular on some occasions. In that case it can be replaced with $\mathbf{G}^T\mathbf{G} + \epsilon\mathbf{I}$ using a small ϵ to ensure the existence of the inverse.

3.3.1 Optimizing widths and locations

The locations and widths has a major role in the quality of the fitting, so it would be desirable to relax the rather artificial constraints given above. Optimizing the average fitting error, is a non-supervised and relatively fast way to find good parameter values. The optimization methods, however, require gradient that exists in closed form since all the functions in (1) are analytical.

To derive the gradient, we first define the error functional. Squared fitting error of all the functions $h = 1, \ldots, N$ can be written as,

$$\begin{aligned} E &= \frac{1}{2}\sum_{h=1}^{N} \left(\mathbf{G}\mathbf{w}_h - \mathbf{y}_h\right)^T \left(\mathbf{G}\mathbf{w}_h - \mathbf{y}_h\right) \\ &= \frac{1}{2}\sum_{h=1}^{N} \left(\mathbf{w}_h^T\mathbf{G}^T\mathbf{G}\mathbf{w}_h - 2\mathbf{y}_h^T\mathbf{G}\mathbf{w}_h + \mathbf{y}_h^T\mathbf{y}_h\right) \end{aligned}$$

The columns of $\mathbf{G}$ are denoted as $\mathbf{G}_k = [\varphi_k(x_1), \varphi_k(x_2), \ldots, \varphi_k(x_m)]^T$ and it's derivative with respect to t_k and σ_k,

$$\begin{aligned} \mathbf{G}_k^{(t)} &= \left[\frac{x_1 - t_k}{\sigma_k^2}\varphi_k(x_1), \ldots, \frac{x_m - t_k}{\sigma_k^2}\varphi_k(x_m)\right]^T \\ \mathbf{G}_k^{(\sigma)} &= \left[\frac{(x_1 - t_k)^2}{\sigma_k^3}\varphi_k(x_1), \ldots, \frac{(x_m - t_k)^2}{\sigma_k^3}\varphi_k(x_m)\right]^T, \end{aligned}$$

respectively. With this notation we obtain,

$$\begin{aligned}\frac{\partial}{\partial t_k}(\mathbf{y}_h^T\mathbf{G}\mathbf{w}_h) &= \mathbf{y}_h^T\mathbf{G}_k^{(t)}w_{h,k}\\ \frac{\partial}{\partial t_k}(\mathbf{w}_h^T\mathbf{G}^T\mathbf{G}\mathbf{w}_h) &= 2\mathbf{w}_h^T\mathbf{G}^T\mathbf{G}_k^{(t)}w_{h,k},\end{aligned}$$

which finally yields,

$$\frac{\partial E}{\partial t_k} = \sum_{h=1}^{n}\left(\mathbf{G}\mathbf{w}_h-\mathbf{y}_h\right)^T\mathbf{G}_{\mathbf{k}}^{(\mathbf{t})}w_{h,k}.$$

Following similar steps for $\partial/\partial\sigma_k$ we get,

$$\frac{\partial E}{\partial \sigma_k} = \sum_{h=1}^{n}\left(\mathbf{G}\mathbf{w}_h-\mathbf{y}_h\right)^T\mathbf{G}_{\mathbf{k}}^{(\sigma)}w_{h,k}.$$

When the gradient is known, the locations and the widths are optimized using unconstrained non-linear optimization. Actually, the problem is constrained because σ cannot be negative. But the kernel (2) is an even function with respect to σ so negative values can be accepted as well. In this paper a Broyden-Fletcher-Goldfarb-Shanno (BFGS) Quasi-Newton method with line search is used [4]. BFGS method resembles quadratic algorithms, such as Newton method, in the sense that it assumes that the problem is quadratic, but with BFGS there is no need to actually compute the Hessian matrix at any stage.

It should be noted, however, that the optimization of the fitting is not equivalent to running a grid search as described above. With the grid search, the prediction performance is optimized directly, while the goal in here is only a good data fitting. Still, a good fitting is a key for a good prediction, since the prediction error cannot be smaller than the fitting error.

4 Experiments

The different function fittings were experimented with the regularly sampled ESTSP'07 benchmark data. First 465 values were used as a learning set and all the parameters were optimized using Leave-One-Out error (LOO). The remaining 410 data points were used as test set to evaluate the quality of the obtained models. In all the tests only the first 15 predicted values were taken into account.

The tests were run for Wavelet, B-spline and Gaussian fitting. The wavelet families Daubechies {1,2,3,4}, Biorthogonal {1.3,2.2} and discrete Meyer were tested. For B-splines orders 2, 3 and 4 were experimented. The Gaussian kernels were tested with fixed kernel locations and grid-search optimized width (as explained in Section 3.3) but also with the Quasi-Newton optimization. The maximum amount of Quasi-Newton iterations was 100. For a reference, a plain k-NN prediction test was carried out. The neighborhood parameter k ranged from 1 to 15 in all the cases.

The experimented window lengths L varied from 15 to 35. Number of coefficients q ranged from 5 to 29. In the case of fixed Gaussian kernels, the width parameter σ got values from 0.3 to 0.93 with 0.07 step size[1].

For comparison, the effect of input normalization was experimented for all the tests. And finally, input selection was tested by running an exhaustive search. For this purpose the models that gave the best LOO error for each number of coefficients (q) were selected. However, due to a rapid increase in computational load only values $q \leq 11$ were experimented.

5 Results

The results are presented in Table 1. The best setup is 2nd order B-spline with input normalization. Functional approach improves results compared to plain k-NN prediction. B-splines give the best overall results and LOO errors are less than one third compared to plain k-NN. The Gaussian fitting is slightly worse.

Wavelet results, on the other hand, are quite similar to plain k-NN. This may be due to the fact that dimensionality is not reduced. k-NN prediction tends to become more difficult as dimensionality grows.

Normalization is beneficial only for the B-splines. With plain k-NN there is not much difference, while in the case of wavelets it is clearly disadvantageous since LOO error becomes roughly 6 fold.

In the case of Gaussian fitting, the fixed kernels with one width parameter perform better. However, the Quasi-Newton method seem to be more robust since test errors are smaller.

5.1 Input Selection

The exhaustive search results are presented in Table 2. Performance is slightly worse compared to the last results, because number of coefficients were restricted to be less than 12. Due to high dimensionality wavelets were left outside of this test. Based on the previous results, inputs were normalized only for B-splines.

Surprisingly the B-spline results were exactly the same because all the inputs were chosen. This may explain the good performance in the previous test; all the coefficients carry relevant information. There was a slight improvement with the Gaussian fitting. But even in this case almost all inputs were selected; Only 9th input out of 11 was omitted for the fixed kernels and 1st, 9th and 10th for Quasi-Newton.

Based on these results, exhaustive search is not recommended because the benefits are minor compared to the increase in computational load.

[1]The width was normalized so that value 1 stands for the distance between the centers

6 Conclusions

Three function spaces were experimented for 15-step ahead prediction of the ESTSP'07 benchmark data. The best setup was 2nd order B-splines with normalized inputs. Gaussian fitting was quite as good while wavelets performed clearly worse.

The results support the notion that the choice of function space is not trivial in FDA. The fact that normalization was beneficial in some occasions stress the problems related to k-NN; input scaling can have a remarkable effect on the performance.

Functional approach improved performance compared to direct prediction method. Moreover, all the methods presented here can be modified to work with irregularly sampled data, which is out of the scope of the traditional time series prediction methods.

References

[1] I. Daubechies, *CBMS-NSF Regional Conference Series in Applied Mathematics vol.61: Ten Lectures on Wavelets*, Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[2] C. de Boor, *Applied Mathematical Sciences vol.27: A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[3] C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, Clarendon, 1995

[4] M. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear Programming, Theory and Algorithms*, John Wiley and Sons, New York, 1993.

[5] J. Ramsay and B. Silverman. *Functional Data Analysis*, Springer Series in Statistics, Springer Verlag, 1997.

[6] D.S. Broomhead, D. Lowe, Multivariable functional interpolation and adaptive networks, *Complex Systems*, 2:321-355, Complex Systems Publication, 1988

[7] P.W. Gaffney, M.J.D. Powell, Optimal interpolation. In G.A. Watson, editor, *On Numerical Analysis*, Lecture Notes in Mathematics 506, pages 90-99, Springer, Heidelberg, 1976.

[8] A. Sorjamaa, N. Reyhani, A. Lendasse, Input and Structure Selection for k-NN Approximator. In proceedings of IWANN 2005, LNCS 3512, pages 985-991, Springer-Verlag, 2005.

[9] A. Lendasse, F. Corona, J. Hao, N. Reyhani, M. Verleysen, Determination of the Mahalanobis matrix using nonparametric noise estimations. In proceedings of ESANN 2006, pages 227-232, April 26-28, Bruges (Belgium), 2006.

[10] T. Kärnä, F. Rossi, A. Lendasse, LS-SVM functional network for time series prediction. In proceedings of ESANN 2006, pages 473-478, April 26-28, Bruges (Belgium), 2006.

B-splines			Normalized	
Degree	LOO	Test	LOO	Test
2	0.0082	0.2004	* **0.0079**	**0.2004**
3	0.0086	0.2088	**0.0080**	**0.2017**
4	0.0105	* 0.1921	**0.0103**	* **0.1921**
Wavelets			Normalized	
Type	LOO	Test	LOO	Test
Daubechies 1	0.0331	0.2189	0.1815	0.4451
Daubechies 2	0.0414	0.1897	0.2933	0.2537
Daubechies 3	0.0393	0.1948	0.2978	**0.1675**
Daubechies 4	0.0342	0.1948	0.2446	* **0.1609**
Biorthogonal 1.3	0.0325	0.1834	0.1712	0.2323
Biorthogonal 2.2	0.0335	0.1852	0.2680	**0.1851**
Discrete Meyer	0.0328	0.2082	* **0.0313**	0.2665
Gaussian			Normalized	
Type	LOO	Test	LOO	Test
Grid search	0.0110	0.1999	* **0.0093**	0.2029
Quasi-Newton	0.0155	* 0.1820	0.0163	0.1958
Plain k-NN			Normalized	
	LOO	Test	LOO	Test
	0.0328	* 0.2082	* **0.0313**	* **0.2082**

Table 1: Results for 15 step ahead prediction. The values are normed mean square errors. The best values for each function type are marked with an asterisk. The cases were normalization improved the results are in bold face.

B-splines					
Input Selection	Degree	LOO	Test	L	q
Yes	2	0.0102	0.1893	30	11
No	2	0.0102	0.1893	30	11
Yes	3	0.0087	0.1901	34	10
No	3	0.0087	0.1901	34	10
Gaussian					
Input Selection	Type	LOO	Test	L	q
Yes	Quasi-Newton	**0.0145**	0.2142	35	11
No	Quasi-Newton	0.0173	0.2088	35	11
Yes	Grid-Search	**0.0138**	0.2304	30	11
No	Grid-Search	0.0140	0.1936	30	11

Table 2: Input selection results. An exhaustive search was run to best models with $q = 5, \ldots, 11$. Improvements are in bold face. For B-splines all the inputs were selected.

Long-Term Prediction of Time Series using a Parsimonious Set of Inputs and LS-SVM

Jarkko Tikka and Jaakko Hollmén

Helsinki University of Technology, Laboratory of Computer and Information Science, P.O. Box 5400, FI-02015 TKK, Finland
e-mail: tikka@mail.cis.hut.fi, Jaakko Hollmen@tkk.fi

Abstract. Time series prediction is an important problem in many areas of science and engineering. We investigate the use of a parsimonious set of autoregressive variables in the long-term prediction task using the direct prediction approach. We use a fast input selection algorithm on a large set of autoregressive variables for different direct predictors, and train non-linear models (LS-SVM and a committee of LS-SVM) on the parsimonious set of non-contiguous set of autoregressive variables. Results will be shown for the time series competition task.

1 Introduction

Time series prediction aims at predicting future values of a time series based on a recorded, finite collection of time series samples. Time series prediction has many applications in natural sciences, engineering, and economics [2, 5, 11].

In this work, we perform input selection of autoregressive variables to produce a parsimonious, or sparse set of input variables for the prediction problem. The resulting set of variables is typically a non-contiguous set of variables, underlying the importance of a few relevant variables that will produce a good predictor. Inputs of the sparse models are selected from a large set of autoregressive input variables for a given past horizon [8, 9]. In addition, the reduction of input space dimension helps us to avoid the problems caused by the curse of dimensionality [10]. On the basis of the selected variables, we train a non-linear predictor, since the linear models do not produce sufficiently accurate predictions in many applications. Here, we have chosen to use the least squares support vector machine (LS-SVM) [7], since it is relatively fast to train and it has only two tuning parameters in the case of Gaussian kernels. The LS-SVM has been successfully used in the time series prediction context, for instance in [6].

In addition to selecting the best model based on cross-validation, we train several perturbed models around the optimum, and give the average prediction of the set of models as our final prediction. This approach takes into account that the tuning parameters are evaluated in predefined grids and small changes in their values could lead even better prediction performance.

The article is organized as follows. The Section 2 describes our approach to long-term time series prediction. In Section 3, the two phase modeling strategy is described. We briefly present the data provided by the conference organizers and present the empirical experiments in Section 4 followed by summary and conclusions in Section 5.

2 Long-Term Prediction of Time Series

In a time series prediction problem, future values of time series are predicted using the previous values. In the long-term prediction, or multi-step-ahead prediction, recurrent or direct approaches can be used. In the recurrent prediction, the previous predictions are used as inputs, thus errors may accumulate and diminish the quality of predictions. In this work, we rely on direct prediction strategy, where the model

$$\hat{y}_{t+k-1} = f_k(y_{t-1}, y_{t-2}, \ldots, y_{t-l}) \tag{1}$$

is used for k-step-ahead prediction. The predicted values are not used as inputs at all in this approach, thus the errors in the predicted values are not accumulated into the next predictions. However, when all the values from y_t to y_{t+k-1} need to be predicted, k different models must be constructed. This increases the computational complexity, but more accurate results are achieved using the direct than the recursive strategy as shown in [8]. In addition, the direct approach allows us to select different input variables to various k-step-ahead prediction models.

3 Two Phase Prediction Approach

3.1 Phase I: Input Selection

Consider the regression problem that there are N measurements available from an output variable y and input variables $x_i, i = i, \ldots, l$. In this phase, the dependency is assumed to be linear

$$y_j = \sum_{i=1}^{l} \beta_i x_{j,i} + \varepsilon_j, \quad j = 1, \ldots, N \ . \tag{2}$$

The errors ε_j are assumed to be independently normally distributed with zero mean and common finite variance. All the variables are assumed to have zero mean and unit variance, thus the constant term is dropped out from the model. The ordinary least squares (OLS) estimates of the regression coefficients $\hat{\beta}_i$ are obtained by minimizing the mean squared error (MSE).

Usually, the quality of the model (2) can be improved by selecting the input variables. Firstly, the generalization ability of the model may increase if only a subset of input variables are used. Secondly, the final model highlights the most important dependencies better and the underlying model is easier to understand or interpret. In this work, we use a previously proposed efficient input selection procedure [8], [9]. The procedure starts by estimating the linear model using all the available inputs. The sampling distributions of OLS estimates $\hat{\beta}_i$ and the standard deviation s_{tr} of the training MSEs are estimated using M times k-fold cross-validation. Mk estimates of each coefficient β_i formulate the sampling distributions. The median m_{β_i} is used as the location parameter for

the distribution, since it is a reasonable estimate for skewed distributions and distributions with outliers. The width of the distribution of $\hat{\beta}_i$ is evaluated using the difference $\Delta_{\beta_i} = \hat{\beta}_i^{high} - \hat{\beta}_i^{low}$, where $\hat{\beta}_i^{high}$ is $Mk(1-q)$th and $\hat{\beta}_i^{low}$ is Mkqth value in the ordered list of the Mk estimates $\hat{\beta}_i$ [3] and q can be set, e.g., $q = 0.165$. With this choice of q, the difference Δ_{β_i} is twice as large as the standard deviation in the case of the normal distribution. The difference Δ_{β_i} describes well the width of both asymmetric and symmetric distributions.

The ratio $|m_{\beta_i}|/\Delta_{\beta_i}$ is used as a measure of significance of the corresponding input variable. The input with the smallest ratio is dropped out from the set of inputs. After that, the cross-validation procedure using the remaining inputs and pruning is repeated as long as there are variables left in the set of inputs.

In the end, we have l different models. The purpose is to select a model which is as sparse as possible, but it still has comparable prediction accuracy. The final model is the least complex model whose validation error is under the threshold $E_v^{min} + s_{tr}^{min}$, where s_{tr}^{min} is the standard deviation of training MSE of the model having the minimum validation error E_v^{min}.

Advantages of the used algorithm is the ranking of the inputs according to their explanatory power and sparseness of the final model. In addition, it is applicable in the case of large number of inputs, since the computational complexity is linear $\mathcal{O}(l)$ with respect to the number of available inputs l.

3.2 Phase II: Non-linear modeling using LS-SVM

Although the linear models are easy to interpret and fast to calculate they are not accurate enough in some problems. Our proposal is to use the selected inputs also in the non-linear model. Goals of this approach are to avoid the curse of dimensionality, over-parameterization, and over-fitting in the non-linear modeling phase. In addition, the interpretability of the non-linear model increases, since only a subset of inputs is included to the model. In this work, we choose the least squares support vector machines (LS-SVM) as a nonlinear predictor [7]. In the primal space, the LS-SVM model is defined as

$$\hat{y} = \boldsymbol{w}^T \phi(\boldsymbol{x}) + b, \tag{3}$$

where $\phi(\cdot)$ is the mapping to the high dimensional feature space. Given a data set $\{\boldsymbol{x}_j, y_j\}_{j=1}^N$, the optimization problem is formulated in the primal space as follows

$$\begin{aligned} \min \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{\gamma}{2}\sum_{j=1}^{N} e_j^2 \\ \text{s.t} \quad & y_j = \boldsymbol{w}^T\phi(\boldsymbol{x}_j) + b + e_j, \quad j = 1, \ldots, N, \end{aligned} \tag{4}$$

where γ is the regularization parameter. In practice, the problem (4) is solved in the dual space and the resulting model is

$$y(\boldsymbol{x}) = \sum_{j=1}^{N} \alpha_j K(\boldsymbol{x}, \boldsymbol{x}_j) + b, \tag{5}$$

where α_j are Lagrange multipliers and the kernel trick $K(\boldsymbol{x}_l, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_l)^T \phi(\boldsymbol{x}_j)$ is applied. The Gaussian kernels $K(\boldsymbol{x}_l, \boldsymbol{x}_j) = \exp(-||\boldsymbol{x}_l - \boldsymbol{x}_j||^2/\sigma^2)$ are used in this work. In this case we have two tuning parameters, which are selected using cross-validation. The LS-SVM model is presented in detail, for example, in [7].

From the input selection point of view, the two phase approach is a so called filter approach [4], since a linear model is used to select a parsimonious set of inputs and a non-linear predictor is trained on top of them. In the earlier work [9], we also performed sensitivity analysis on the selected input variables in order to investigate the importance of variables in non-linear model. Since the competition setting does not give too much weight on the interpretation, we omit this aspect from this work.

4 Experiments

In this section, we present our empirical experiments with the competition data set provided by the symposium organizers. The data set is briefly described in the Section 4.1, model selection in Section 4.2 and the results are shown in the section 4.3.

4.1 Prediction competition data set

Time series competitions are a good way to measure out-of-the sample generalization ability of predictors; they have a long history [11] and have become a relatively popular setting for comparing methods for time series predictions. In this competition, the domain of origin of the data remains so far completely unknown to the competitors.

The data set provided by the organizers of the conference is a scalar time series $y_t,\ t = 1, \ldots, N$ with $N = 875$ samples measured in a discrete time grid. The goal of the competition is two-fold: both the 15 and 50 next predicted values of the time series will be used to measure the goodness of the prediction approach. The accuracy of the predictors will be measured by the MSE between the predicted values and the true values, which are not accessible by the competitors. The goodness measure has then the form with $K = 15, 50 : MSE_K = \frac{1}{K}\sum_{k=1}^{K}(y_{t+k-1} - \hat{y}_{t+k-1})^2$. The MSE measures an overall performance averaged over the whole prediction horizon. The short-term prediction being supposedly easier and resulting in smaller MSE values.

4.2 Model selection with cross validation

In order to generalize well, the prediction must be accurate not only in the training set, but also in a data set that is not part of the training. We have left out the last 50 samples from the original data set, which is used as a test set. This simulates the competition setup. In order to estimate generalization properties of the predictors, we have used rest of the data in training and validation. All the models are trained using normalized (zero mean and unit variance) data, but all the results, for instance the errors, are shown in the original scale.

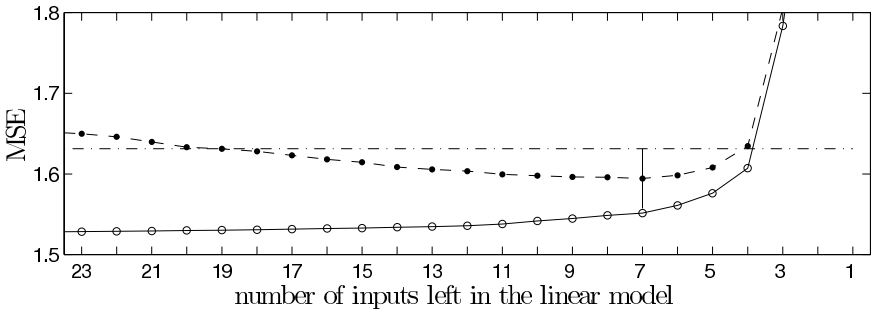

Fig. 1: Illustration of input selection in the prediction of y_{t+24}, training error (*solid line with circles*) and validation error (*dashed line with dots*) as a function of the number of inputs in the linear model. The vertical line marks the minimum validation error and the horizontal dash-dotted line represents the threshold, which is used in the selection of the final model.

In the input selection, 10-fold cross-validation repeated $M = 100$ times is used. This choice produces 1000 estimates for the coefficients β_i, which is considered to be large enough for reliably estimating the distribution of the parameters in the linear model. The LS-SVM models using selected inputs are evaluated using 15 values of the regularization parameter γ and 15 values of the kernel parameter σ^2, which are logarithmically equally spaced in the ranges $\gamma \in [10^{-2}, 10^5]$ and $\sigma^2 \in [10^{-3}, 10^4]$. The optimal values of γ and σ^2 are selected using 10-fold cross-validation repeated ten times to increase the reliability of the results.

For the final predictions used in the competition, we used all the available data to train the models with the parameters producing minimum validation errors. These models cannot obviously be validated against the available data set anymore.

4.3 Results

As a result of our approach to predict the time series, we get a parsimonious set of inputs, which provides insight to the system itself. Since this is of little value in the competition setting, we concentrate on the prediction results.

The maximum number of inputs is set to be $l = 50$, i.e. the available inputs are $y_{t-l}, l = 1, \ldots, 50$ in each of the 50 prediction cases. Figure 1 illustrates how the training and the validation errors develop as a result of our input selection procedure in the case of 25-step-ahead (y_{t+24}) prediction. It is notable that the validation error does not increase drastically until almost all the inputs are dropped out from the model. If the final model had been selected according to the minimum validation error the number of inputs would have been 7. However, even more parsimonious model is achieved when the thresholding is used. The

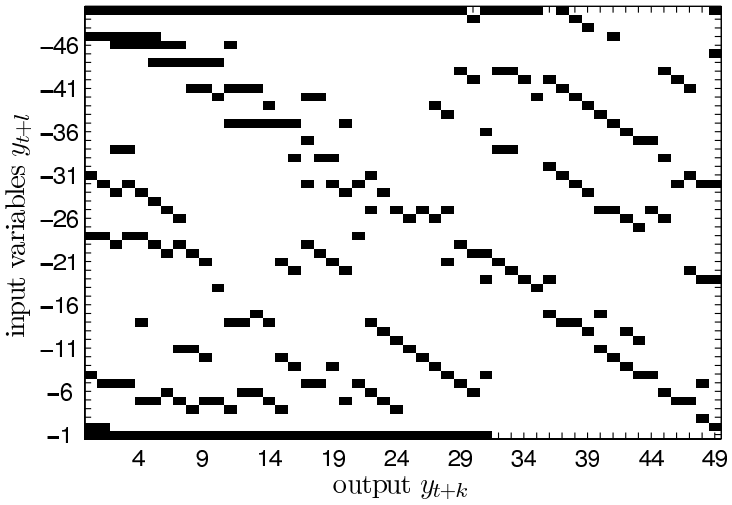

Fig. 2: The selected inputs in the final models. The outputs $y_{t+k}, k = 0, 1, \ldots, 49$ are in the x-axis and the available inputs $y_{t+l}, l = -1, -2, \ldots, -50$ are in the y-axis. The selected inputs are denoted by black rectangles in each column.

final number of inputs is 5 and the validation error does not increase significantly.

In Figure 2, we illustrate the selected inputs in our direct prediction tasks with prediction horizon varying from 1 to 50 ($y_t, \ldots, y_{t+49}$). The black squares in each of the vertical columns denote the inclusion of the variable to the prediction model. The figure shows that each of the direct predictors has a different set of variables. The first available autoregressive input variable y_{t-1} is interestingly included in all the models for predicting y_{t+k-1} upto the value of $k = 32$. Also, the last autoregressive input y_{t-50} is included in most of the models. The partial diagonal patterns indicate the inclusion of a variable with the same absolute lag difference from the value to be predicted. It is noteworthy, that the pattern of selected input variables is relatively sparse, approximately only 5 inputs are chosen from the available 50 autoregressive input variables ($y_{t-1}, \ldots, y_{t-50}$). For instance, the selected 5 inputs in 25-step-ahead prediction model (y_{t+24}) are y_{t-1}, y_{t-4}, y_{t-12}, y_{t-27}, and y_{t-50}. The direct predictor will then have the form $\hat{y}_{t+24} = f_{24}(y_{t-1}, y_{t-4}, y_{t-12}, y_{t-27}, y_{t-50})$ and the form of the prediction function f_{24} is left to be specified.

In Figure 3, the validation mean squared errors for direct k-step-ahead predictions $y_{t+k-1}, k = 1, ...50$ are shown. The dashed line with circles indicates the mean squared errors of the linear predictors using the parsimonious, or sparse pattern of autoregressive input variables. The solid line with dots indicates the mean squared errors of the LS-SVM models trained using the same inputs as the linear models. The non-linear predictors (LS-SVM) show consistently bet-

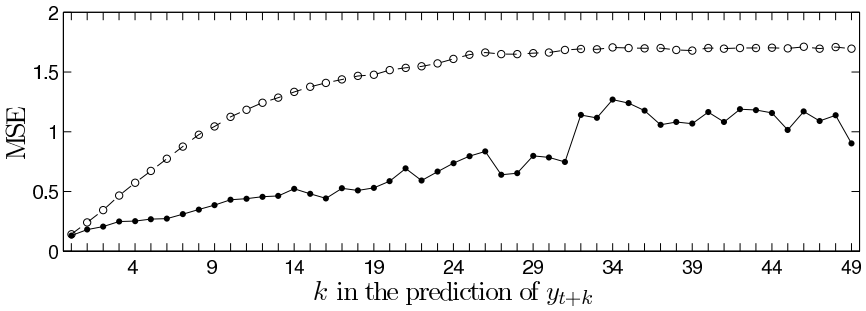

Fig. 3: The mean-square error (MSE) calculated for the validation set for different direct predictors $y_{t+k-1}, k = 1, \ldots, 50$. The MSE for the sparse linear model has been plotted with a dashed line marked with circles, the MSE for the LS-SVM has been plotted with a dotted solid line.

ter performance over whole prediction horizon in the validation set. The errors increase when the prediction horizon increases, which is expected behavior.

In Figure 4, the absolute errors of predictions $|y_{t+k-1} - \hat{y}_{t-k+1}|, k = 1, \ldots, 50$ are shown for the linear models (dashed line with circles) and for the LS-SVM models (solid line with dots) in the test set. The LS-SVM model has smaller absolute error in 19 cases out of the 50, but it only performs clearly better in the prediction of 19-22 steps-ahead. On the other hand, the linear model is clearly more accurate in the prediction of 9-11 and 34-46 steps-ahead.

In order to improve the performance of the LS-SVM models we train a committee of LS-SVM models for each k-step-ahead prediction. It is known that combining the models to form a committee can significantly improve the predictions on new data [1]. For each k-step-ahead prediction case we train a committee, which consists of 10 LS-SVM models. To get variation for the members of committees we train them as follows. Firstly, we train each member using a random subsample of data whose size is 9/10 of the original data. Secondly, the kernel parameters for the members of committee are linearly equally spaced in the range $\sigma^2 \in [\sigma^2_{opt} - \sigma^2_{opt}/2, \sigma^2_{opt} + \sigma^2_{opt}/2,]$, where σ^2_{opt} is the optimal value obtained using cross-validation. The same value of regularization parameter γ is used for each member of the committee. Obviously, the σ^2_{opt} and γ might vary in different k-step-ahead prediction models. In the end, the output of the committee is a mean of the outputs of the members of the committee.

The mean-square test errors $MSE_K = \frac{1}{K}\sum_{k=1}^{K}(y_{t+k-1} - \hat{y}_{t+k-1})^2, K = 15, 50$ for the linear models, the LS-SVM models, and committees of LS-SVMs are presented in Table 1. Although the cross-validation has been done carefully, the test set (the last 50 points not part of the validation or the training set) indicates worse performance for the LS-SVM models than for the linear models. The committees of LS-SVM do not either perform as well as the linear models. This is clearly a contrary finding with the results from the validation results.

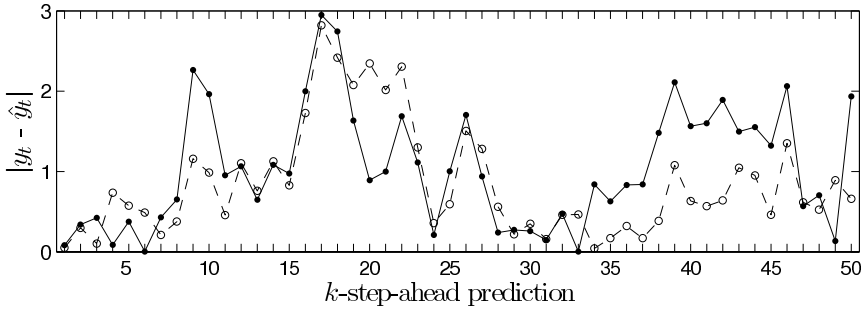

Fig. 4: The absolute errors of predictions $|y_{t+k-1} - \hat{y}_{t-k+1}|, k = 1, \ldots, 50$ in the test set. Dashed line with circles and solid line with dots indicate the errors in the linear and LS-SVM models, respectively.

MSE	sparse linear	LS-SVM	LS-SVM committee
15-step-ahead	0.51	0.98	0.90
50-step-ahead	1.18	1.63	1.60

Table 1: The averaged MSEs for linear models, LS-SVM models, and committees of LS-SVM models for 15- and 50-step-ahead prediction in the test set.

Certainly, Figure 3 indicates the superiority of the LS-SVMs over linear models in the validation set. At this time, we cannot attribute this phenomenon to any particular cause. However, we noted that the averaged MSEs are largely influenced by a small fraction of unsuccessful predictions in the evaluation of test errors in Table 1. Thus, these results can be considered as a precautionary example how well-validated methods can perform poorly in a small test set. In this setup, we have only one sample for each k-step-ahead prediction model in the test set.

The given predictions that will be our contribution to the prediction competition are illustrated in the Figure 5. The upper panel indicates the predictions of the linear predictors with the selected inputs. The lower panel illustrates the predictions using LS-SVM models using selected inputs (black dashed line) and committee of LS-SVMs (gray dashed line). The values on left from the vertical black line are the input values, i.e last 50 values of given time series. No final conclusions can be drawn about quality of the predictions, since the actual values are unknown. However, the predictions using linear model are obviously smoother than predictions using non-linear models. Especially, the time steps from 908 to 920 may be poorly predicted by the non-linear methods. It is also noteworthy, that there are almost no differences between the predictions of LS-SVMs and the committees of LS-SVMs.

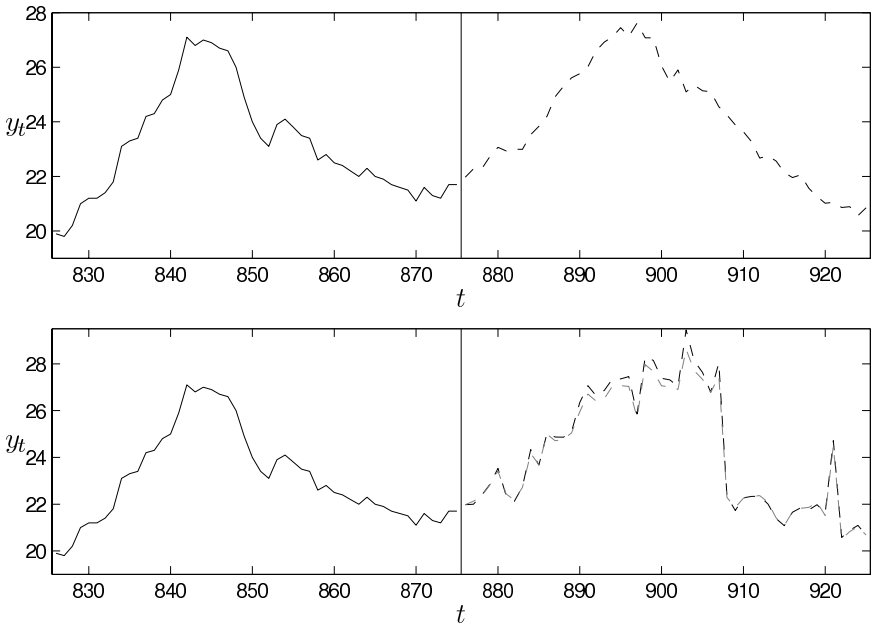

Fig. 5: The predicted values for the purpose of the competition. All 50 predicted values are shown, the predictions are given by the sparse linear model (*upper panel*), LS-SVM (*lower panel, black dashed line*) and a committee of LS-SVM models (*lower panel, gray dashed line*).

5 Summary and Conclusions

Parsimonious sets of input variables provide many advantages in time series prediction. For instance, the final model requires less parameters to be trained. Also the sparsity provides a good basis for interpretation, and highlights the relevant dependencies in time series. In a time series prediction competition setting, we have selected parsimonious inputs in the spirit of backward selection, and trained a LS-SVM prediction model based on the selected inputs. As a final attempt to improve the prediction accuracy, we have trained a set of models on slightly perturbed parameters around the optimal cross-validated model parameters and averaged the results. The goal of our approach is to produce accurate results, and at the same time provide comprehensible views into the system through the set of parsimonious regressors.

Acknowledgments

This work has been supported by the Academy of Finland, the grant number 116853.

References

[1] Christopher Bishop. *Neural Networks in Pattern Recognition*. Oxford Press, 1996.

[2] Chris Chatfield. *Time Series Forecasting*. Chapman & Hall/CRC, 2002.

[3] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993.

[4] Keinosuke Fukunaga. *Introduction to Statistical Pattern Recognition*. Computer Science and Scientific Computing. Academic Press, second edition edition, 1990.

[5] James D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

[6] A. Lendasse, V.Wertz, G.Simon, and M. Verleysen. Fast bootstrap applied to LS-SVM for long term prediction of time series. In *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks*, pages 705–710, 2004.

[7] Johan A.K. Suykens, Tony Van Gestel, Jos De Brabanter, Bart De Moor, and Joos Vandewalle. *Least Squares Support Vector Machines*. World Scientifc Publishing, 2003.

[8] Jarkko Tikka, Jaakko Hollmén, and Amaury Lendasse. Input Selection for Long-Term Prediction of Time Series. In Joan Cabestany, Alberto Prieto, and Francisco Sandoval, editors, *Proceedings of the 8th International Work-Conference on Artificial Neural Networks (IWANN 2005)*, volume 3512 of *Lecture Notes in Computer Science*, pages 1002–1009. Springer-Verlag, 2005. Vilanova i la Geltú, Barcelona, Spain.

[9] Jarkko Tikka, Amaury Lendasse, and Jaakko Hollmén. Analysis of fast input selection: Application in time series prediction. In Stefanos Kollias, Andreas Stafylopatis, Wlodzislaw Duch, and Erkki Oja, editors, *Proceedings of the 16th International Conference on Artificial Neural Networks (ICANN'06)*, volume 4132 of *Lecture Notes in Computer Science*, pages 161–170. Springer-Verlag, 2006.

[10] Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In Joan Cabestany, Alberto Prieto, and Francisco Sandoval, editors, *Proceedings of the 8th International Work-Conference on Artificial Neural Networks (IWANN 2005)*, volume 3512, pages 758–770. Springer-Verlag, 2005.

[11] A. S. Weigend and N. A. Gershenfeld, editors. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, 1994.

Time Series Prediction of Mira Light Curve using CNLS Neural Network based on AAVSO Data

Hossein Peyvandi[1, 2] Ahmad Dalaki[2] Caro Lucas[3] *

1- Faculty of Scientific Applied Telecommunication,
P.O.Box: 19395-1544, Tehran - IRAN
Email: hpeyvandi@gmail.com
2- Sciences and Astronomical Center of Tehran
No.11, Aarafat St., Niavaran, Tehran - IRAN
Email: ahmad@dalaki.com
3- Control and Intelligent Processing Center of Excellence
Tehran University - IRAN
Email: lucas@ipm.ir

Abstract. Some stars vary in brightness with respect to time. Plot of the brightness versus time is called Light Curve. A star named Mira in the constellation Cetus fluctuates in brightness with a long period, about 11 months. There are several different methods of time series prediction and analysis which can be applied to variable stars light curves. We used a neural network named: CNLS (Connectionist Normalized Local [Linear] Spline) that is an extension of RBF neural network to predict of Mira light curve future values based on 10,000 AAVSO (American Association of Variable Stars Observers) data.

Keywords: Mira, Light Curve, Astronomy, CNLS, Neural Network, AAVSO, RBF, Prediction, Linear Spline

1 Introduction

It is possible for individual stars to vary in brightness all by themselves. Thousands of such stars are known in astronomy. The periods of the variations can be seconds for some stars or years for others. Plots of the brightness in magnitude versus time are called Light Curves [1]. One type of the variable stars named Mira variables after its brightest member, Mira star (in constellation Cetus). Mira itself may appear sometimes in magnitude 9, and is thus invisible for naked eye. In this research, we considered Mira star light curve data as time series which it is found in AAVSO website [2].

Time series modeling by neural networks is known as intelligent modeling [3, 4]. The advantages of intelligent modeling are: robustness, better fitting and simplicity of use. Among neural networks: MLP, RBF and some of their extensions have been used widely in modeling researches and applications.

In this paper, we have shown the predictive ability of the CNLS neural network. The model based on CNLS which has been prototyped is a contribution to observatory team.

* The work was supported in part by Sciences and Astronomical Center of Tehran (SACT)

2 Theory

In this section it is shown how we are able to model a time series behavior and use estimated model to prediction tasks.

2.1 Takens Theorem

If we suppose Δ as time lag and m is an integer value then:

$$x_{t+p} = f(x_t, x_{t-\Delta}, x_{t-2\Delta}, \cdots x_{t-(m-1)\Delta}) \tag{1}$$

Where p is predicted time and f is a smooth function. Takens theorem says: "**upper limit for m to predict of x at t+p is equal to 2d+1 for any desired approximation where d is fractal dimension of time series attractor**" [5]. Then we are sure with satisfied data of past values we can predict future values of known time series.

2.2 Neural Networks

Neural networks due to their universal approximation property constitute a good solution for modeling complex functions, and then we may use them to make a map of function f in (1) [6].

2.2.1 CNLS: Topology

CNLS neural network is considered as RBF extension which is shown in Figure 1 with one output for prediction tasks.

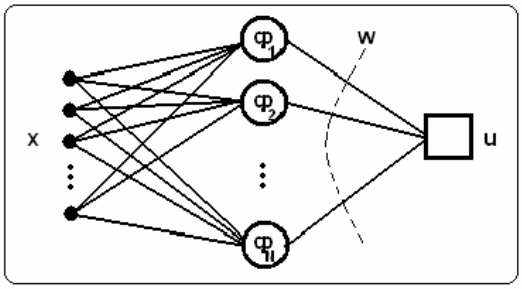

Fig. 1: CNLS Neural Network Topology with Single Output

Suppose x as input vector, w as weighting vector then u is output and ϕ_i is kernel function for i-th neuron as:

$$\phi_i(x) \stackrel{\Delta}{=} \exp[-0.5(\frac{x - m_i}{\sigma_i})^2] \tag{2}$$

In CNLS, kernel function is similar to RBF.

2.2.2 CNLS: Learning Rules

Equations (3) show iterated equations to update the Weights (w) and Coefficients (d) vectors, in brief:

$$w_i^{p+1} = w_i^p + \mu_w[d(x_p) - u(x_p)].\frac{\phi_i(x_p)\sum_{k=1}^{N}\phi_k(x_p)}{\sum_{k=1}^{N}\phi_k^2(x_p)} \quad (3.a)$$

$$d_i^{p+1} = d_i^p + \mu_d[d(x_p) - u(x_p)].\frac{(x_p - m_i)\phi_i(x_p)\sum_{k=1}^{N}\phi_k(x_p)}{\sum_{k=1}^{N}\left|(x_p - m_i)^2\phi_k^2(x_p)\right|} \quad (3.b)$$

Where: $\mu_w, \mu_d \leq 1/3$ and random initial values for vectors [3].

Indeed, CNLS differs from the RBF in two ways:
- Kernel function Normalization, and
- Addition of a linear term, due to Taylor expansion about x.

3 Experimental Results

3.1 Preprocessing

In this research, we have analyzed over 10,000 visual measures of Mira made by the AAVSO during the past 35 years as raw data and then with mean average of adjacent irregular samples, we have constructed a regular sampled time series.

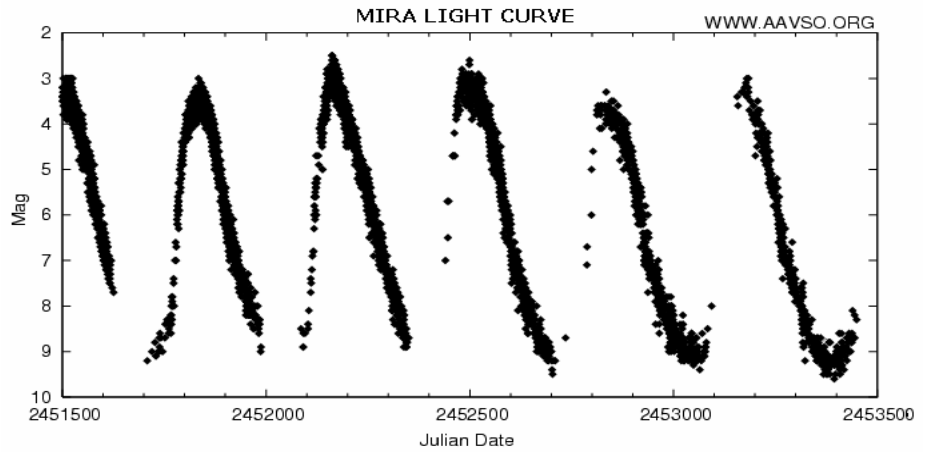

Fig. 2: Mira Light Curve for JD2451500 to JD2453500. (The last 2000 days are shown)

Irregularity in astronomy observations is occurred often as it is seen in Figure 2. Julian Date can be considered a very simple calendar, where its calendar date is just an integer. This is useful for reference and conversions.

3.2 CNLS as Predictor

3.2.1 Structures

We used several structures of CNLS to train over regulated samples of Mira light curve time series for prediction future values ahead with p=1,12,25. CNLS (30-17-1) resulted the best among 10 different structures which it had 30 inputs and 17 neurons with single output. Besides, we considered MLP (30-11-7-1) with 2 hidden layers and trained it same as for CNLS and compared the results.

3.2.2 Predictions

Figure 3 shows the CNLS outputs for p=1, 12, 25. Only 1200 samples (about 3.5 periods) are shown. CNLS shows its ability for prediction, at least for small p values.

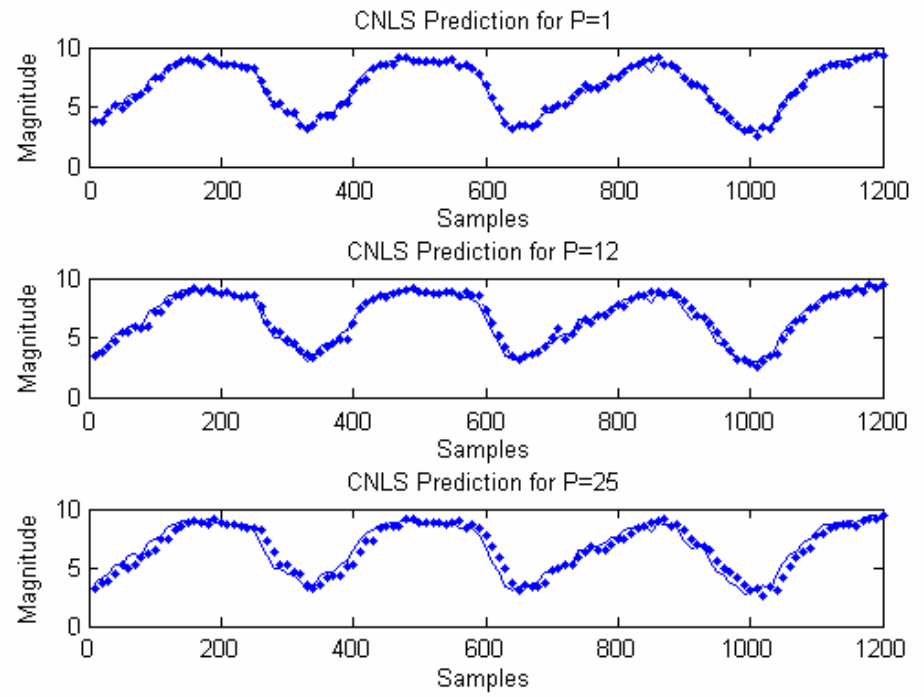

Fig. 3: CNLS (30-17-1) Predictions for p=1, 12, 25.

3.3 CNLS vs. MLP

In Figure 4, it is shown the learning curves for CNLS and MLP in our experiment when p=1. All results are compared after 10000 epochs. In this figure LSN is considered as:

$$LSN(dB) = 10\log_{10}[MSE/(x_{\max} - x_{\min})^2] \quad (4)$$

We can say CNLS is a fast learner vs. MLP and it gives us a reasonable result after few epochs. Instead, MLP may achieve the better result if learning continues.

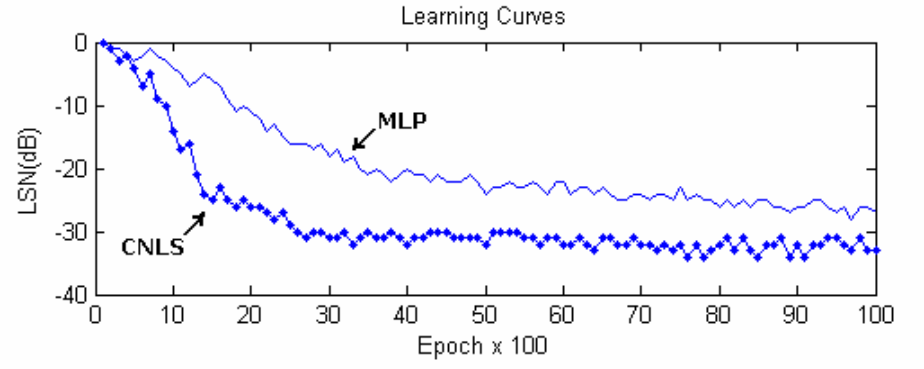

Fig. 4: Learning Curves for CNLS and MLP (p=1).

In Table 1 it is shown the final (after 10000 epochs) results for both CNLS and MLP. Error in predicted magnitude is about 0.2 for -33dB.

P	CNLS(30-17-1)	MLP(30-11-7-1)
1	-33	-27
12	-25	-24
25	-17	-20

Table 1: CNLS and MLP Final Results (LSN Values) for p=1, 12, 25.

4 Conclusions

In this paper, CNLS neural network used for prediction of Mira light curve time series and all results for 3 different predicted values compared with MLP ones. CNLS was a very fast learner in adding, it used few computational resources. MLP needed to huge resources for learning and its results had no promising advantages except for long-term prediction. CNLS short-term prediction of Mira light curve was excellent.

Using new features [7] to train CNLS, new methods [7, 8] for this research and also comparison with other methods is our current study.

5 Acknowledgements

We acknowledge with thanks the variable star observations from the AAVSO International Database contributed by observers worldwide and used in this research.

The first author wishes to thank Dr. B. Raastini for helpful discussions on early version of this research and for his valuable comments.

References

[1] J. M. Pasachoff, Astronomy: From the Earth to the Universe, 4th ed., Saunders College Publishing, ISBN: 0-03-031329-5, 1991.

[2] AAVSO website: http://www.AAVSO.org

[3] H. Peyvandi, Intelligent Modeling of Time Series, M.Sc Thesis, Supervisor: M. R. Asharif, Advisor: C. Lucas, Tehran University, 1995.

[4] J. Hertz et al, Introduction to the Theory of Neural Computation, Addison-Wesley, ISBN: 0-201-50395-6, 1991.

[5] F. Takens, Detecting Strange Attractor in Turbulence, Lecture Notes in Mathematics, D. Rand, L. Young (editors), Springer Berlin, PP366-381, 1981.

[6] A. Lapedes and R. Farber, Nonlinear Signal Processing using Neural Networks: Prediction and System Modelling, Technical Report, LA-UR-87-2662, Los Alamos National Lab., 1987.

[7] H. Peyvandi, B. Fazaeefar, H. R. Amindavar, "Determining Class of Underwater Vehicles using Hidden MARKOV Model with HAUSDORFF Similarity Measure," *proceeding of IEEE Oceanic Engineering Society Symposium on Underwater Technology*, UT' 98, pages 258-261 , April 15-17 ,Tokyo, JAPAN, 1998.

[8] J. S. Albus, A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC), Trans. ASME, Journal of Dynamic Systems, Vol.97, PP220-227, 1975.

Long-term prediction of chaotic time series via spectral analysis and neurofuzzy models

Masoud Mirmomeni[1], Caro Lucas[2], and Elahe Ahmadi[3]

1- Control and Intelligent Processing Center of Excellence, Electrical and Computer Eng. Department, University of Tehran, Tehran, Iran
2- School of Cognitive Sciences, Institute for studies in theoretical Physics and Mathematics, Tehran, Iran
3- Sharif University of Technology, Electrical Eng. Department, Tehran, Iran

m.mirmomeni@ece.ut.ac.ir, lucas@ipm.ir, ahmadi@ee.sharif.edu

Abstract. In this paper, a method based on spectral analysis and neurofuzzy modeling is proposed that is capable of issuing very accurate long term prediction of natural chaotic phenomena. A locally linear neurofuzzy model is optimized for each of the principal components obtained from singular spectrum analysis, and the multi step predicted values are recombined to make the natural chaotic phenomena. As real world case studies, the method has been applied to the long-term prediction of Darwin sea level pressure and sunspot number time series. Results depict the power of the proposed method in long-term prediction of chaotic time series.

Keywords. Long-term prediction, chaotic time series, singular spectrum analysis, locally linear neurofuzzy model.

1 Introduction

The neural and neurofuzzy models, as general function approximators [1, 2], have performed well in the prediction of nonlinear and chaotic time series [3, 4]; but when the number of observations for training is limited; they can neither reconstruct the dynamics nor can learn the shape of attractor. They may present the most accurate one step ahead predictions, but in larger prediction horizon their performance dramatically falls down.

In this paper a decomposition method based on singular spectrum analysis is used to make an intuitive nonlinear black box modeling technique applicable to long term prediction of natural time series. Singular Spectrum Analysis (SSA) is originally designed to extract information from short noisy chaotic time series, to provide an insight to the unknown dynamics and to enhance the signal to noise ratio [5, 6]. In addition, SSA performs a data adaptive filtering in the lag coordinate space of data and yields the principal components of time series which have narrow band frequency

spectra and obvious temporal patterns. The principal components include linear or nonlinear trends, periodic and quasi periodic patterns and some lower amplitude signals which can be considered as colored noise. Most of the narrow band periodic components can be estimated via simple and optimal linear models, while there are always some more complex patterns which present nonlinear characteristics. Thus in reconstructing the original time series from the principal components, one should use both linear and nonlinear techniques and also the linearity tests.

The locally linear neurofuzzy models [1] have a key advantage in this situation; with an incremental tree based learning algorithm the model starts as an optimal linear least squares estimation, and the nonlinear neurons are added if they result in an enhancement in performance. Thus the learning algorithm eliminates the need for linearity tests, and automatically constructs the model to achieve the highest generalization. The locally linear neurofuzzy model can be used as a general framework to predict the main patterns of the time series. The components obtained from SSA, most of which have linear or simple nonlinear behaviors, are long term predictable, and long term prediction of the original time series is accessible via the fourth stage of SSA by recombining the extrapolated components.

The paper consists of six sections. The mathematical description of SSA is presented in section 2. Section 3 presents the main aspects of the locally linear neurofuzzy model. Section 4 is devoted to describe the learning method used for locally linear neurofuzzy models to predict chaotic time series. Prediction of three well known nonlinear case studies has been considered in section 5. Long-term prediction of the selected time series is very difficult due to their chaotic characteristics and positive Lyapunov exponents. Darwin sea level pressure time series and sunspot number have been considered as difficult real world case studies in this section, where long term prediction of the 22nd and 23rd solar cycles has been presented and compared to the predictions made by physical precursor and solar dynamo techniques. The last section contains the concluding remarks.

2 Spectral analysis

SSA is defined as a new tool to extract information from short and noisy chaotic time series [6]. It relies on the Karhunen-Loeve decomposition of an estimate of covariance matrix based on *M* lagged copies of the time series. Thus as the first step, the embedding procedure is applied to construct a sequence $\{\tilde{X}(t)\}$ of *M*-dimensional vectors from time series $\{X(t): t = 1,\ldots,N\}$

$$\tilde{X}(t) = (X(t), X(t+1),\ldots, X(t+M-1)), \quad t = 1,\ldots,N', \quad N' = N - M + 1 \tag{1}$$

The $N' \times M$ trajectory matrix (*D*) of the time series has the *M* dimensional vectors as its columns, and is obviously a Hankel matrix (the elements on the diagonals *i+j=const* are equal). In the second step, the $M \times M$ covariance matrix C_X is calculated as

$$C_X = \frac{1}{N'} D^T D \tag{2}$$

and its eigenelements can be determined by Singular Value Decomposition (SVD)

$$C_X = U \Sigma V^T \quad ; \quad U^T U = I \ , \ V^T V = I \tag{3}$$

The elements of diagonal matrix $\Sigma = [diag(\sigma_1, \ldots, \sigma_M)]$ are the singular values of D and are equal to square roots of the eigenvalues of C_X. The eigenelements $\{(\lambda_k, \rho_k) : k = 1, \ldots, M\}$ of C_X are obtained from

$$C_X \rho_k = \lambda_k \rho_k \tag{4}$$

Each eigenvalue, λ_k estimates the partial variance in the direction of ρ_k, and the sum of all eigenvalues equals the total variance of the original time series. In the third step, the time series is projected onto each eigenvector, and yields the corresponding principal components

$$A_k(t) = \sum_{j=1}^{M} X(t + j - 1) \rho_k(j) \tag{5}$$

Each of the principal components, being a nonlinear or linear trend or a periodic or quasi-periodic pattern, has narrow band frequency spectra and well defined characteristics to be estimated. As the fourth step, the time series is reconstructed by combining the associated principal components

$$R_K(t) = \frac{1}{M_t} \sum_{k \in K} \sum_{j=L_t}^{U_t} A_k(t - j + 1) \rho_k(j) \tag{6}$$

The normalization factor (M_t), and the lower (L_t) and upper (U_t) bounds of reconstruction procedure differ for the center and edges of the time series, and are defined by the following formula

$$(M_t, L_t, U_t) = \begin{cases} \left(\frac{1}{t}, 1, t\right), & 1 \le t \le M - 1 \\ \left(\frac{1}{M}, 1, M\right), & M \le t \le N' \\ \left(\frac{1}{N - t + 1}, t - N + M, M\right), & N' + 1 \le t \le N \end{cases} \tag{7}$$

To enhance signal to noise ratio, one can use the singular spectrum plot (the logarithmic scale plot of singular values of covariance matrix in decreasing order). The principal components related to lower singular values can be omitted in reconstruction stage to obtain adaptive noise cancellation. If all the components are used in reconstructing the time series, no information is lost.

3 Neuro-fuzzy modeling

The fundamental approach with locally linear neuro-fuzzy (LLNF) model is dividing the input space into small linear subspaces with fuzzy validity functions. Any produced linear part with its validity function can be described as a fuzzy neuron.

Thus the total model is a neuro-fuzzy network with one hidden layer, and a linear neuron in the output layer which simply calculates the weighted sum of the outputs of locally linear models (LLMs).

$$\hat{y}_i = \omega_{i_0} + \omega_{i_1} u_1 + \omega_{i_2} u_2 + \ldots + \omega_{i_p} u_p \tag{8}$$

$$\hat{y} = \sum_{i=1}^{M} \hat{y}_i \phi_i(\underline{u}) \tag{9}$$

where $\underline{u} = \left[u_1 \, u_2 \, \cdots \, u_p \right]^T$ is the model input, M is the number of LLM neurons, and ω_{ij} denotes the LLM parameters of the ith neuron. The structure of LLNF is shown in Fig. 1.

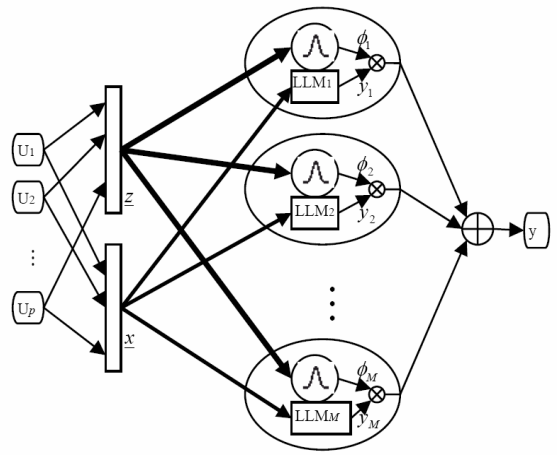

Fig. 1: Structure of locally linear neuro-fuzzy model

The validity functions are chosen as normalized Gaussians; normalization is necessary for a proper interpretation of validity functions.

$$\phi_i(\underline{u}) = \frac{\mu_i(\underline{u})}{\sum_{j=1}^{M} \mu_j(\underline{u})} \tag{10}$$

$$\mu_i(\underline{u}) = \exp\left(-\frac{1}{2} \left(\frac{(u_1 - c_{i1})^2}{\sigma_{i1}^2} + \ldots + \frac{(u_p - c_{ip})^2}{\sigma_{ip}^2} \right) \right) = \exp\left(-\frac{1}{2} \frac{(u_1 - c_{i1})^2}{\sigma_{i1}^2} \right) \times \ldots \times \exp\left(-\frac{1}{2} \frac{(u_p - c_{ip})^2}{\sigma_{ip}^2} \right) \tag{11}$$

Each Gaussian validity function has two sets of parameters, centers (c_{ij} s) and standard deviations (σ_{ij} s) which are the $M.p$ parameters of the nonlinear hidden layer. Optimization or learning methods are used to adjust both the parameters of

local linear models (ω_{ij} s) and the parameters of validity functions (c_{ij} s and σ_{ij} s). Global optimization of linear parameters is simply obtained by Least squares technique. The complete parameter vector contains $M.(p+1)$ elements:

$$\underline{\omega} = \lfloor \omega_{10} \quad \omega_{11} \quad ... \quad \omega_{1p} \quad \omega_{20} \quad \omega_{21} \quad ... \quad \omega_{M0} \quad ... \quad \omega_{Mp} \rfloor \tag{12}$$

and the associated regression matrix $\underline{X}$ for N measured data samples is

$$\underline{X} = [\underline{X}_1 \quad \underline{X}_2 \quad ... \quad \underline{X}_M] \tag{13}$$

$$\underline{X}_i = \begin{bmatrix} \phi_i(\underline{u}(1)) & u_1(1)\phi_i(\underline{u}(1)) & \dots & u_p(1)\phi_i(\underline{u}(1)) \\ \phi_i(\underline{u}(2)) & u_1(2)\phi_i(\underline{u}(2)) & \dots & u_p(2)\phi_i(\underline{u}(2)) \\ \vdots & \vdots & & \vdots \\ \phi_i(\underline{u}(N)) & u_1(N)\phi_i(\underline{u}(N)) & \dots & u_p(N)\phi_i(\underline{u}(N)) \end{bmatrix} \tag{14}$$

Thus

$$\hat{\underline{y}} = \underline{X}.\hat{\underline{\omega}} \quad ; \quad \hat{\underline{\omega}} = \left(\underline{X}^T \underline{X}\right)^{-1} \underline{X}^T \underline{y} \tag{15}$$

The remarkable properties of locally linear neuro-fuzzy model, its transparency and intuitive construction, lead to the use of least squares technique for rule antecedent parameters and incremental learning procedures for rule consequent parameters, the advantages of which are considered in the next section.

4 Learning methodologies

The most important property of locally linear neuro-fuzzy model is that one can use some intuitive algorithms in training. The model starts as an optimal least squares estimation, and the new local linear models are created to reduce the prediction error. In each iteration the worst performing locally linear neuron is determined to be divided. All the possible divisions in the *p* dimensional input space are checked and the best is performed. The splitting ratio can be simply adjusted as $\frac{1}{2}$, which means that the locally linear neuron is divided into two equal halves on the selected input dimension. Based on such a division the centers (c_{ij}) and standard deviations (σ_{ij}) of the new neurons are computed and the fuzzy validity functions for the new structure are updated according to the equations 3 and 4. The center of validity functions are the centers of the new hyper-cubes, and the standard deviations are usually set as 0.7. The algorithm is as follows:

1. *The initial model:* start with a single locally linear neuron, which is a globally optimal linear least squares estimation over the whole input space with $\Phi_1(\underline{u}) = 1$, and $M = 1$.
2. *Find the worst neuron*: Calculate a local loss function e.g. MSE for each of the $i = 1,\dots,M$ locally linear neurons, and find the worst performing neuron.
3. *Check all divisions*: The worst neuron is considered for further refinement. The validation hypercube of this neuron is divided into two halves with an axis orthogonal split. Divisions in all dimensions are tried, and for each of the *p* divisions the following steps are carried out:

a. Construction of the multi-dimensional validity functions for both generated hyper cubes.
b. Local estimation of the rule consequent parameters for both newly generated neurons.
c. Calculation of the total loss function or error index for the current overall model.

4. *Validate the best division*: The best of the p alternatives in step 3 is selected. If it results in reduction of loss functions or error indices on training and validation data sets, the related validity functions and neurons are updated, the number of neurons is incremented $M = M + 1$, and the algorithm continues from step 2, otherwise the learning algorithm is terminated.

This automatic learning algorithm provides the best linear or nonlinear model with maximum generalization, and performs well in prediction applications. The error index used in the experiments of this study is Normalized Mean Square Error (NMSE), which is defined as

$$NMSE = \left(\frac{\sum_{i=1}^{n} (y - \hat{y})^2}{\sum_{i=1}^{n} (y - \bar{y})^2} \right) \tag{16}$$

where y, $\hat{y}$, and $\bar{y}$ are observed data, predicted data and average of observed data respectively. This algorithm can be implemented to yield a locally linear neurofuzzy model for each component of a chaotic time series which is given by SSA and nonlinear trend of such time series could be modeled by combining the output of each neurofuzzy model.

5 Case studies

5.1 Long-term prediction of Darwin sea level pressure time series

Among the most commonly used and analyzed sets of climate time series are the El niño indices such as the Southern Oscillation Index (SOI) [7, 8]. In this case study, long-term prediction of Darwin Sea Level Pressure (DSLP) as one of the SOI indexes is considered. The DSLP consists of 1400 data (from 1890 to present). 800 data is used for training and the next 600 samples are used to test the long-term prediction.

Fig. 2 shows the singular spectrum of DSLP time series. The window length is 30 and the first 8 components, related to the first 8 singular values, are chosen to reconstruct the time series. The incremental learning algorithm constructs the locally linear model automatically and the embedding dimension of each component is defined by an autocorrelation analysis. A locally linear model is trained for each of the components and is then used in long term prediction by reconstructing the long term trend of the time series via long term prediction of each component which have long term predictable trends. The result of 15 steps ahead predicting of the Darwin sea level pressure index using LLNF with SSA approach is presented in Fig. 3.

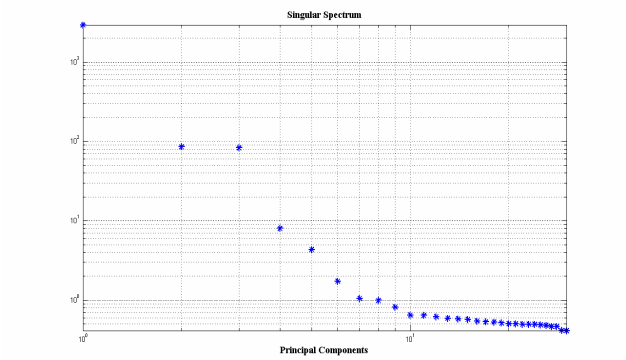

Fig. 2: Singular spectrum of Darwin Sea Level Pressure time series by SSA algorithm

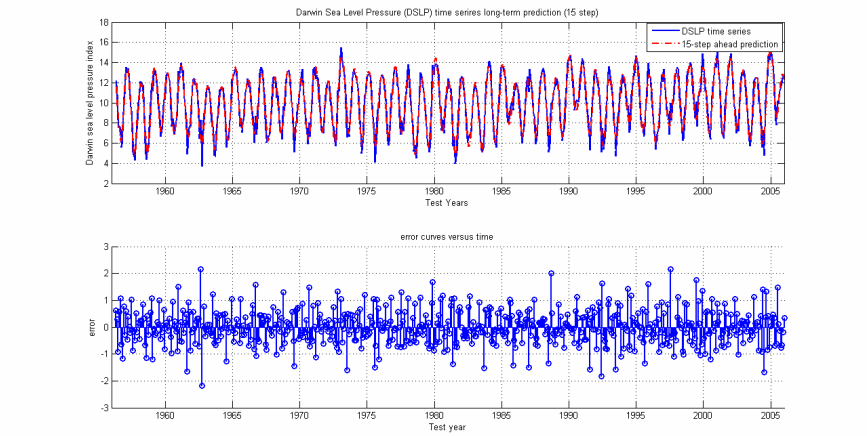

Fig. 3: Fifteen step ahead prediction of DSLP time series by the proposed method; (Upper) solid: the real time series values, dashed line: prediction via proposed method; (Lower) Fifteen step ahead prediction error

5.2 **Long-term prediction of solar activity**

The level of sun's activity, defined by the occurrence of solar flares, coronal mass ejections, and sunspots, has quasi periodic variations with a period of about eleven years. Each eleven-year solar cycle starts with a period of quiescence called solar minimum, and gradually turns into a period of activity called solar maximum. A good measure of activity level is sunspot number, which has been saved since the early 18th century. Long term prediction of solar activity is of great importance due to its harm to satellites and space missions. The numerical prediction techniques including the most powerful neural and neurofuzzy models with appropriate learning methodologies [9, 10, and 11], however can perform the most accurate one year ahead

predictions, cannot be used in long term predictions. As a result of this fact, the physical precursor and solar dynamo methods are preferred for long term solar activity forecasting [12, 13, and 14].

The solar cycle is an excellent case study for the method proposed in this article. In this case study, the first 12 components, related to the first 12 singular values, are chosen to reconstruct the time series. These components are shown in Fig. 4 and the others, with maximum amplitude of 3, are eliminated to enhance the signal to noise ratio. A locally linear model is trained for each of the components and is then used in long term prediction. This method is used in predicting the current solar cycle (cycle 23). The observations to 1995 are used for training, and the multi step predicted values are shown in Fig. 5. The predicted value is 132 which is very close to the actual value (120). Table 1 presents a comparison between this method and the predictions made by physical precursor and solar dynamo techniques. Using all the available data of sunspot number time series results in an early prediction of the peak of next solar cycle; the maximum international smoothed sunspot number is predicted to be near 145 in late 2011.

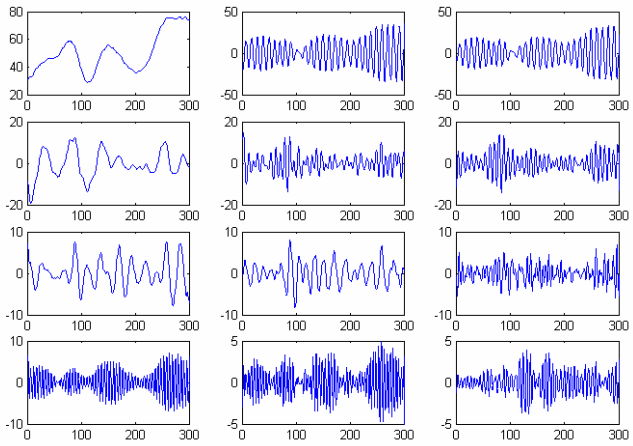

Fig. 4: The first twelve principal components of sunspot number time series by SSA algorithm; left to right then top to bottom: PC1 to PC12.

	Time of prediction	Predicted value for the maximum
Precursor	1996	160 ± 30
Solar Dynamo	1993	170 ± 25
Solar Dynamo	1995	138 ± 30
Solar Dynamo	1998	143 ± 30
The proposed method	1995	132
The actual value	------	120

Table 1: Several methods in long term prediction of solar cycle 23 via the smoothed peak of sunspot number time series (the observed value is 120 in 2000)

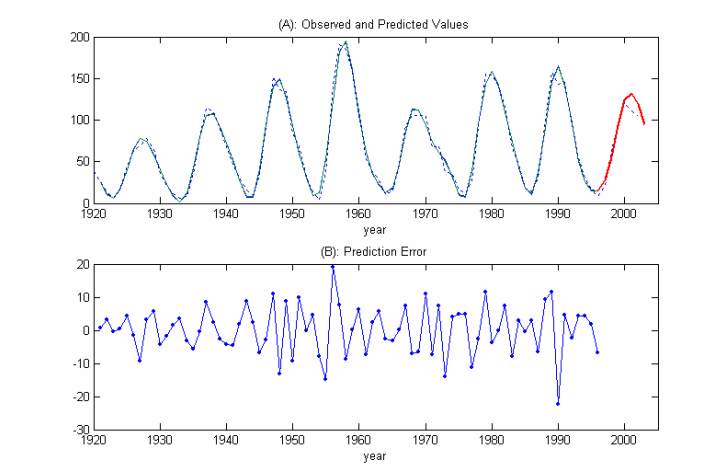

Fig. 5: Long term prediction of solar cycle 23 by the proposed method; (A) dotted: the observed values, solid line from 1920 to 1996: the reconstructed and one step ahead predicted values, and thick line from 1996 to 2003: long term predicted values; (B) one step ahead prediction error

6 Discussion

Long term prediction of natural phenomena with limited number of observations is usually difficult. In this research a combination of singular spectrum analysis and locally linear neurofuzzy modeling is proposed to improve the capability of automatic black box modeling techniques for long term prediction. The principal components of natural time series, obtained from SSA, have narrow band frequency spectra and definite linear or nonlinear trends and periodic patterns. The main patterns of these components are long term predictable and the incremental learning algorithm for the locally linear neurofuzzy model, constructs the best linear or nonlinear model with highest generalization for each of the components. The original time series and its long term prediction are obtained by recombining the multi step predicted components. The proposed method has been primarily tested on two difficult case studies from chaotic systems, and has been successfully used in solar activity forecasting. The accuracy of predictions has been initially validated via the neurofuzzy modeling of components, and the reconstructed results for solar cycles 22 and 23 are compared to the well-known physical precursor and solar dynamo techniques and show the usefulness of the proposed method.

References

[1] O. Nelles, editor. Unsupervised *Nonlinear System Identification*, Springer Verlag, Berlin, 2001.

[2] S. Haykin, editor. Unsupervised *Neural networks: A Comprehensive Foundation*, Macmillan, New York, 1994.

[3] B. Lillekjendlie, D. Kugimutzis and N. Christophersen, Chaotic time series, part II: System identification and prediction, *Identification and Control*, 15: 225-243, 1994.

[4] H. Leung, T. Lo, and S. Wang, Prediction of noisy chaotic time series using an optimal radial basis function neural network, *IEEE Transactions on Neural Networks*, 12(5): 1163-1172, 2001.

[5] R. Vautard and M. Ghil, Singular spectrum analysis in nonlinear dynamics with applications to paleoclimatic time series, *Physica D*, 35: 395-424, 1989.

[6] R. Vautard P. Yiou, and M. Ghil, Singular spectrum analysis: a toolkit for short noisy chaotic signals, *Physica D*, 58: 95-126, 1992.

[7] A. Kaplan, Y. Kushnir, and M. A. Kane, Reduced space optimal interpolation of historical of marine sea level pressure: 1854-1992, *Journal of Climate*, 13: 2987-3002, 2000.

[8] R. Mendelssohn, S. J. Bograd, F. B. Schwing, and D. M. palacios, Teaching old indices new tricks: A state-space analysis of El Niño related climate indices, *Geophysical Research Letter*, 32: L07709, doi: 10.1029/2005GL022350, 2005.

[9] O. Uluyol, M. Ragheb, and S. R. Ray, Local output gamma feedback neural network, proceeding of the *IEEE Int. Conf. on Neural networks: IJCNN*, 1: 337-342, 1998.

[10] C. Lucas, A. Abbaspour, A. Gholipour, B. N. Araabi, and M. Fatourechi, Enhancing the performance of neurofuzzy predictors by emotional learning algorithm, *Informatica*, 27(2): 165-174, 2003.

[11] A. Gholipour, A. Abbaspour, , B. N. Araabi and C. Lucas, Enhancements in the prediction of solar activity by locally linear model tree, proceeding of *MIC2003: 22nd Int. Conf. on Modeling, Identification and Control*, Innsbruck, Austria, 158-161, 2003.

[12] G. M. Brown, The peak of solar cycle 22: prediction in retrospect, *Annales Geophysicae*, 10:453-470, 1992.

[13] R. Thompson, A technique for predicting the amplitude of solar cycle, *Solar Physics*, 148: 383, 1993.

[14] J. A. Jocelyn, Panel achieves consensus prediction of solar cycle, *EOS Tran. AGU*, 78: 211-212, 1997.

Development of LoLiMoT learning method for training neuro-fuzzy models on-line

Masoud Mirmomeni[1], Babak Nadjar Araabi[1, 2] and Caro Lucas[1, 2]

1- Control and Intelligent Processing Center of Excellence, Electrical and Computer Eng. Department, University of Tehran, Tehran, Iran
2- School of Cognitive Sciences, Institute for studies in theoretical Physics and Mathematics, Tehran, Iran

m.mirmomeni@ece.ut.ac.ir, araabi@ut.ac.ir, lucas@ipm.ir

Abstract. Several methods have been introduced for modeling and system identification of different types of processes. Unfortunately, most of these methods have a training phase which should be done offline. This paper develops LoLiMoT learning algorithm as one of the most useful learning methods for system identification of nonlinear systems to an on-line learning algorithm which is called O-LoLiMoT. This method obviates some of the LoLiMoT restrictions in tuning premise parameters of the locally linear models via some clustering approaches. Two case studies are considered. Results depict the power of the proposed method in on-line system identification of nonlinear time-variant dynamic systems.

Keywords. Nonlinear systems, on-line learning, neuro-fuzzy models, clustering, O-LoLiMoT.

1 Introduction

System identification aims at finding a mathematical model from the measurement record of inputs and outputs of a system. To do so, in most proposed methods a collection of input output data is used offline to tune the parameters of the model according to a performance index or a cost function. It is obvious that this produced model is a time invariant model. But when one looks around, there are lots of phenomena which possess time variant behavior which is often caused by aging or wearing of components. Also, in some applications, may the model structure used for modeling a process be too simple in order to be capable of describing that kind of process in all relevant operating regimes with the desired accuracy. Therefore to identify a system, why the parameters of a model should be tuned offline? Although there are lots of methods which tune the parameters of a linear systems on-line [1], [2], and [3] there are few methods for on-line nonlinear system identification and developing ordinary learning algorithms for nonlinear system identification to on-line applications should be a great development [4], [5], [6], and [7].

On the other hand, Locally Linear Neuro-Fuzzy (LLNF) models are particularly well suited for on-line learning since they are capable of solving the so-called *stability/plasticity dilemma* [8]. One of the most popular learning algorithms for

LLNF models is Locally Linear Model Tree (LoLiMoT) algorithm [3] which is an incremental learning algorithm in adjusting the premise and consequence parameters of such models. LLNF models with this incremental learning algorithm were implemented to model several complex systems and the performance of this approach was great in describing complex phenomena [9], [10], [11], and [12]. Unfortunately, this algorithm should be done offline and has some restrictions in its structure. First of all, as it said before this algorithm is a growing and a one way learning algorithm and it is not possible to return to some former state during learning phase. Also, in this algorithm the region which should be divided to two sub-regions, is always divided in to two equal halves. In addition, it has to be said that using LoLiMoT algorithm for on-line learning is difficult to utilize directly since its computational demand grows linearly with the number of training data samples [13], [14].

This paper develops the ordinary LoLiMoT algorithm by proposing a novel learning method for on-line system identification of nonlinear time variant systems. This method is called On-line Locally Linear Model Tree (O-LoLiMoT) algorithm. To obviate the restrictions of the ordinary LoLiMoT algorithm in tuning the premise parameters of LLNF models, this algorithm uses k-means clustering algorithm to tune the parameters of premise parts. To show the performance of this novel learning algorithm for on-line applications, several case studies are considered. Results depict the great performance of this on-line learning algorithm in describing such complex systems.

The remaining of this paper is structured as follows. Section 2 briefly illustrates the main aspects of locally linear neuro-fuzzy models. Section 3 is devoted to describe the on-line learning methodology which is used for locally linear neuro-fuzzy models to model nonlinear time variant system. Section 4 is devoted to show the performance of the proposed learning algorithm in modeling some nonlinear systems. The last section contains the concluding remarks.

2 Neuro-fuzzy modeling

The fundamental approach with locally linear neuro-fuzzy (LLNF) model is dividing the input space into small linear subspaces with fuzzy validity functions. Any produced linear part with its validity function can be described as a fuzzy neuron. Thus the total model is a neuro-fuzzy network with one hidden layer, and a linear neuron in the output layer which simply calculates the weighted sum of the outputs of locally linear models (LLMs).

$$\hat{y}_i = \omega_{i_0} + \omega_{i_1} u_1 + \omega_{i_2} u_2 + \ldots + \omega_{i_p} u_p \tag{1}$$

$$\hat{y} = \sum_{i=1}^{M} \hat{y}_i \phi_i(\underline{u}) \tag{2}$$

where $\boldsymbol{u} = \lfloor\, u_1 u_2 \cdots u_p \,\rfloor^T$ is the model input, M is the number of LLM neurons, and ω_{ij} denotes the LLM parameters of the ith neuron. The validity functions are chosen as normalized Gaussians; normalization is necessary for a proper interpretation of validity functions.

$$\phi_i(\underline{u}) = \frac{\mu_i(\underline{u})}{\sum_{j=1}^{M} \mu_j(\underline{u})} \tag{3}$$

$$\mu_i(\underline{u}) = \exp\left(-\frac{1}{2}\left(\frac{(u_1 - c_{i1})^2}{\delta_{i1}^2} + \ldots + \frac{(u_p - c_{ip})^2}{\delta_{ip}^2}\right)\right) = \exp\left(-\frac{1}{2}\frac{(u_1 - c_{i1})^2}{\delta_{i1}^2}\right) \times \ldots \times \exp\left(-\frac{1}{2}\frac{(u_p - c_{ip})^2}{\delta_{ip}^2}\right) \tag{4}$$

Each Gaussian validity function has two sets of parameters, centers (c_{ij} s) and standard deviations (δ_{ij}) which are the $M \times p$ parameters of the nonlinear hidden layer. Optimization or learning methods are used to adjust both the parameters of local linear models (ω_{ij}) and the parameters of validity functions (c_{ij} and δ_{ij}). Global optimization of linear parameters is simply obtained by Least squares technique. The complete parameter vector contains $M \times (p+1)$ elements:

$$\boldsymbol{\omega} = \lfloor \omega_{10} \quad \omega_{11} \quad \ldots \quad \omega_{1p} \quad \omega_{20} \quad \omega_{21} \quad \ldots \quad \omega_{M0} \quad \ldots \quad \omega_{Mp} \rfloor^T \tag{5}$$

and the associated regression matrix $\underline{X}$ for N measured data samples is

$$\boldsymbol{X} = [\boldsymbol{X}_1 \quad \boldsymbol{X}_2 \quad \ldots \quad \boldsymbol{X}_M] \tag{6}$$

$$\boldsymbol{X}_i = \begin{bmatrix} \phi_i(\underline{u}(1)) & u_1(1)\phi_i(\underline{u}(1)) & \ldots & u_p(1)\phi_i(\underline{u}(1)) \\ \phi_i(\underline{u}(2)) & u_1(2)\phi_i(\underline{u}(2)) & \ldots & u_p(2)\phi_i(\underline{u}(2)) \\ \vdots & \vdots & & \vdots \\ \phi_i(\underline{u}(N)) & u_1(N)\phi_i(\underline{u}(N)) & \ldots & u_p(N)\phi_i(\underline{u}(N)) \end{bmatrix} \tag{7}$$

Thus

$$\boldsymbol{y} = \boldsymbol{X}\hat{\boldsymbol{\omega}} \quad ; \quad \hat{\boldsymbol{\omega}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{8}$$

3 On-Line learning method for LLNF models

This section is devoted to describe the novel on-line learning method for locally linear models based on LoLiMoT learning algorithm and k-means clustering algorithm to identify time variant nonlinear systems. In this algorithm, it is tried to obviate some of the restrictions of the LoLiMoT algorithm.

The LoLiMoT algorithm belongs to the class of *growing* strategies because it incorporates an additional rule in each iteration of the algorithm. During the training procedure some of the formerly made divisions may become suboptimal or even superfluous. However LoLiMoT does not allow one to undo these divisions. In this paper with a *pruning* strategy which is able to merge formerly locally linear models, it

is tried to remedy this drawback. In addition, in this algorithm splitting is always done in a way that the sub-regions be equal halves. This paper uses the k-means clustering algorithm in the proposed algorithm to obviate this restriction. K-means clustering minimizes the following loss function:

$$I = \sum_{j=1}^{C} \sum_{i=1}^{N} \mu_{ji} \left\| \boldsymbol{u}(i) - \boldsymbol{c}_j \right\|^2 \rightarrow \min_{\underline{c}_j} \tag{9}$$

where the index i runs over all elements of the sets S_j, C is the number of clusters, $\underline{c}_j$ are the cluster centers (prototypes) and $\mu_{ji} = 1$ if the sample $\underline{u}(i)$ is associated (belongs) to cluster j and $\mu_{ji} = 0$. The sets S_j contain all indices of those data samples (out of all N) that belong to the cluster j, i.e., which are nearest to the cluster center $\underline{c}_j$. The cluster centers $\underline{c}_j$ are the parameters that the clustering technique varies in order to minimize (9). The k-means algorithm used for tuning premise parameters (validity functions) of LLNF models in the proposed method is modified to be useful for on-line applications. This clustering method works as follows:

- *Assigning current data sample to a proper cluster (locally linear model):* in proposed method, the number of clusters or locally linear models is not known a priori and according to the current data sample it is decided to label a new cluster to the current data sample or assign it to a proper existing cluster via some distance criteria. To do so, the minimum distance between the current data sample and the clusters' prototype are calculated. If this value is more than a threshold value, a new cluster with a prototype as current data sample is created and if it is not, current data sample is assigned to the cluster with nearest cluster prototype.
- *Tuning the center of validity functions:* If current data sample is assigned to a cluster, the centroid (mean) as the prototype of that cluster should be modified. The cluster prototype is modified as

$$\boldsymbol{c}_j = \frac{\sum_{i \in S_j} \boldsymbol{u}(i)}{N_j} \tag{10}$$

  where $\underline{u}(i)$ is the current data sample, and N_j is the number of those data samples that belong to cluster j, and $\underline{c}_j$ is the cluster's prototype. The modification rule has a recursive form to be useful for on-line applications.
- *Tuning the covariance matrix of the validity functions:* in this paper, the validity functions are set to be Gaussian. Therefore, the covariance matrixes of these validity functions should be tuned either. To do so, after tuning the clusters' prototype the covariance matrix of each validity functions is set as follows:

$$\Sigma_j = \frac{1}{3} \times diag\left(\min_{i \neq j} \left(\boldsymbol{c}_j, \boldsymbol{c}_i \right) \right) \tag{11}$$

where Σ_j is the covariance matrix of validity function j (cluster j), and $\boldsymbol{c}_j$ is the cluster prototype of the cluster j, and $\boldsymbol{c}_i$ is the cluster prototype for other clusters.

- *Checking for merging:* this part is considered to remedy the other restriction of the LoLiMoT algorithm in tuning the premise parameters. After tuning the parameters of the cluster's prototype (mean of Gaussians validity functions) and covariance matrixes, it is checked that if some clusters are close enough to consider as a unique cluster. This could be done by calculating the distance between cluster's prototypes. Again, if the distances between some clusters are smaller than a merging threshold, these clusters is merged to a single cluster and the prototype for this new cluster could be calculated as follows:

$$\boldsymbol{c}_j = \frac{\sum_{i=1}^{p} N_i \boldsymbol{c}_i}{\sum_{i=1}^{p} N_i}, \quad p\text{: number of current clusters} \tag{12}$$

$$N_j = \sum_{i=1}^{p} N_i$$

The covariance matrix of this validity function (cluster) could be tuned according to 3.

- *Local estimation of the rule consequent for newly generated clusters:* For an on-line adaptation of the rule consequent parameters the local estimation approach is chosen. Therefore the numerical robustness becomes a critical issue since the number of parameters in global estimation can be very large [3]. To do so, the Weighted Least Square (WLS) algorithm is used for local estimation of the rule consequent parameters for newly generated clusters.

Fig. 3 shows the proposed on-line algorithm "On-line Locally Linear Model Tree algorithm (O-LoLiMoT)" which is used for system identification of some time variant nonlinear dynamic systems in next section.

4 Case studies

4.1 Mackey-Glass time series prediction

The Mackey-Glass system has been introduced as a model of white blood cell production [15]. This time series is produced by a time-delay difference system of the form:

$$\frac{dx(t)}{dt} = \beta x(t) + \frac{\alpha x(t-\tau)}{1 + x^{10}(t-\tau)} \tag{13}$$

where $x(t)$ is the value of the time series at time t. This system is chaotic for $\tau > 16.8$. In this paper, the Mackey-Glass time series is constructed with parameter values $\alpha = 0.2$, $\beta = -0.1$, $\tau = 30$ and $x_0 = 1.5$. Parameter τ is changed two times

during this simulation: in sample 300, the parameter τ is changed to 50 and in sample 800 this parameter is changed to 17. The threshold parameter for splitting in O-LoLiMoT algorithm is set to 0.5 in this case study and the threshold parameter for merging is set to 0.3. Fig. 2 shows the performance of the O-LoLiMoT algorithm in one step ahead prediction of Mackey-Glass chaotic time series. It can be seen that in the beginning, the performance of the model is poor, but after a short time, the

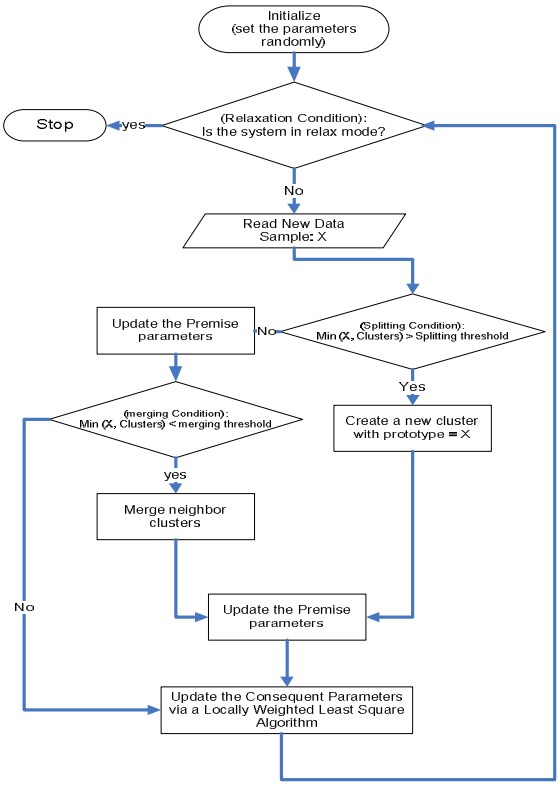

Fig. 1: The O-LoLiMoT algorithm structure for system identification of nonlinear systems.

performance of the model increases rapidly. In addition, it is obvious that during samples 300 and 800, the performance of the model gets a little poor, but after a short time, model learns the new dynamic as well.

Fig. 3 shows the number of clusters (locally linear models) during O-LoLiMoT algorithm's operation. Table 1 presents the results obtained from O-LoLiMoT, MLP and RBF (which are trained on-line). In contrast, LoLiMoT shows noticeably better predictions. It can be easily seen that the result of the LLNF is superior in comparison to the results of the prediction by other methods.

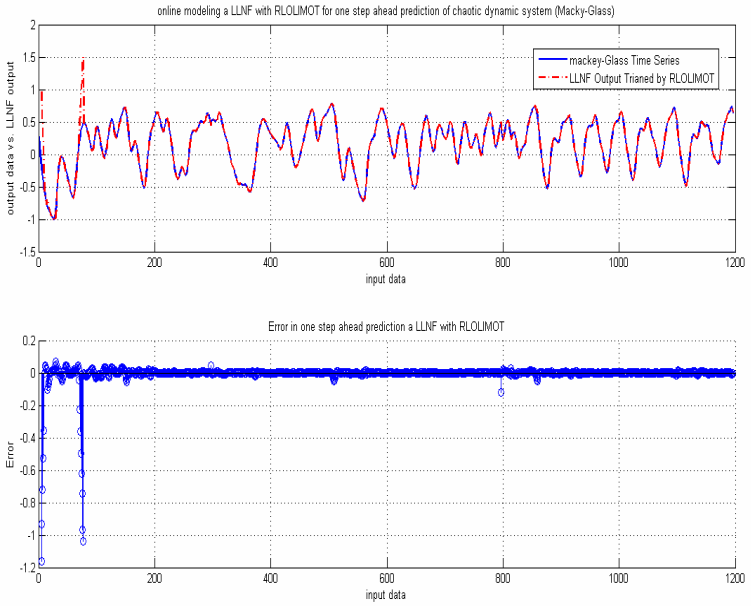

Fig. 2: One step ahead prediction of Mackey-Glass time series by the proposed method; (upper) solid: the real time series values, dashed line: prediction via O-LoLiMoT algorithm; (lower) one step ahead prediction error

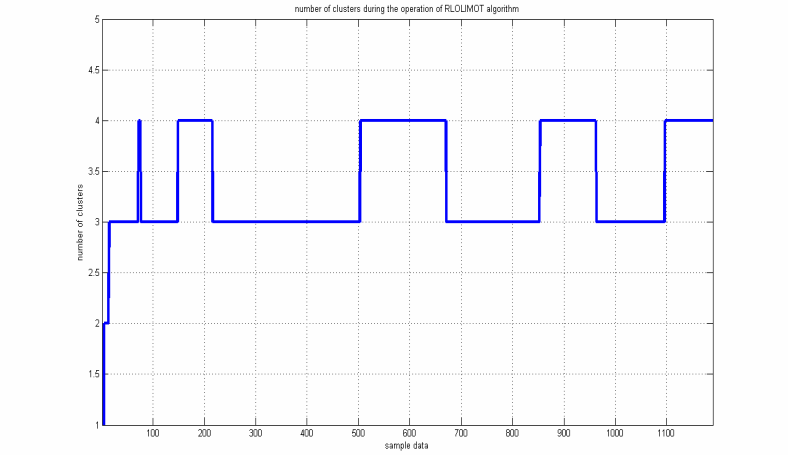

Fig. 3: Number of clusters (locally linear models) during O-LoLiMoT algorithm's operation for one step ahead prediction of Mackey-Glass time series.

Model	MLP	RBF	LLNF
Learning Algorithm	BP	RLS	O-LoLiMoT
Number of neurons	14	20	Often 4
MSE (best fit)	0.0016	0.00102	0.000961

Table 1: Comparison of several methods in predicting time variant Mackey-Glass chaotic time series on-line

4.2 System identification of heat exchanger dynamics as a nonlinear time variant system

This section is devoted to check the performance of the proposed on-line learning method in modeling a heat exchanger dynamics as a nonlinear time variant dynamics. The data set of this dynamics is available on-line [16].

The threshold parameter for splitting in O-LoLiMoT algorithm is set to 1.5 in this case study and the threshold parameter for merging is set to 1. Fig. 4 shows the performance of the O-LoLiMoT algorithm in modeling heat exchanger nonlinear dynamics. It can be seen that in the beginning of the O-LoLiMoT algorithm's operation, the performance of the model is poor, but after a short time, the performance of the model increases rapidly. Fig. 5 shows the number of clusters during O-LoLiMoT algorithm's operation. Table 2 presents the results obtained from O-LoLiMoT, MLP and RBF for system identification of this nonlinear dynamic system.

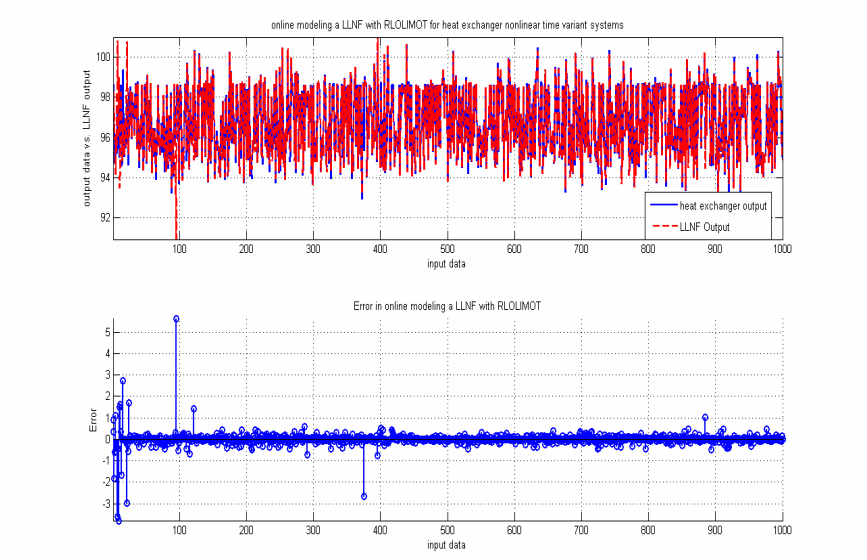

Fig. 4: modeling the heat exchanger nonlinear time variant dynamics by the proposed method; (upper) solid: the heat exchanger output values, dashed line: prediction via O-LoLiMoT algorithm; (lower) modeling error

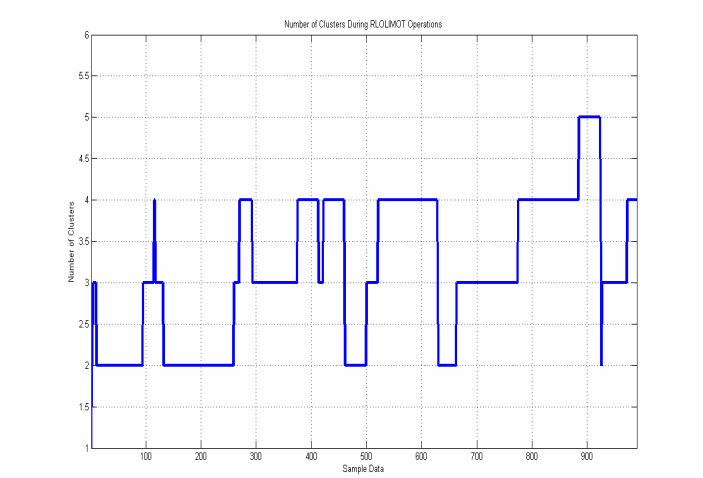

Fig. 5: Number of clusters (locally linear models) during O-LoLiMoT algorithm's operation for modeling heat exchanger nonlinear time variant dynamics.

Model	MLP	RBF	LLNF
Learning Algorithm	BP	RLS	O-LoLiMoT
Number of neurons	21	45	Between 3 and 4
MSE (best fit)	0.048	0.018	0.0031

Table 2: Comparison of several methods in modeling heat exchanger nonlinear time variant dynamics.

5 Conclusions

In this paper, the problem of on-line system identification of nonlinear time variant systems is considered. It has to be said that the literature on-line identification methods of nonlinear system is not as rich as the literature on offline identification methods. Basically, our motivation was the possibility of using LoLiMoT algorithm as an incremental learning algorithm to use its useful characteristics for tuning the parameters of LLNF models. The difficulty was the offline structure of this learning algorithm. So we had to develop this algorithm to on-line applications. To do so, this paper proposes a new learning method called O-LoLiMoT algorithm as an incremental learning algorithm to tune the parameters of LLNF models with the aid of k-means clustering algorithm. In this algorithm some restrictions of LoLiMoT algorithm are obviated. The great performance of LLNF models with O-LoLiMoT learning algorithm in predicting nonlinear time variant dynamics such as Mackey-Glass chaotic time series and modelling heat exchanger system as a nonlinear time variant dynamics depict the potential of proposed learning method in this subject.

References

[1] L. Ljung, *System Identification Theory for the User*, Prentice Hall, Inc. 1999.

[2] G. P. Liu, *Nonlinear Identification and Control, a Neural Network Approach*, Springer Verlag, London, 2001.

[3] O. Nelles, *Nonlinear System Identification*, Springer, Berlin, 2001.

[4] W. Broclanann, O. Huwendiek, Prediction in real-time control using adaptive networks with on-line learning, Proceeding of the 3rd IEEE Conf. on Control applications, Strathclyde University, Glascow, 1067-1072, Aug. 1994.

[5] T. W. Wan, L. C. How, Development of feedback error learning strategies for training neurofuzzy controllers on-line, Proc. of the 10th IEEE International Conference on Fuzzy Systems, Melbourne, Australia, 1016-1021, Dec. 2001.

[6] J. A. Domínguez-López, R. I. Damper, R. M. Crowder and C. J. Harris, Hybrid neurofuzzy on-line learning for optimal grasping, Proc. of the 2nd Int. Conf. on Machine Learning and Cybernetics, Xian, 803-808, 2-5 November 2003.

[7] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar, Learn++: an incremental learning algorithm for supervised neural network, IEEE Trans, on Systems, Man, and Cybernetics—Part C: Applications and Reviews, 31(4): 497-508, Nov. 2001.

[8] G. Carpenter, S. Grossberg, The ART of adaptive pattern recognition by a self-organizing neural network, IEEE Computer, 77-88, Mar. 1988.

[9] M. Jalili-Kharaajoo, R. Ranji, and H. Bagherzadeh, Predictive control of fossil power plant based on local linear model tree (LOLIMOT), Proc. of IEEE, 633-638, 2003.

[10] M. Jalili-Kharaajoo, A. Rahmati, and F. Rashidi, Internal motor control based on local linear model tree (LOLIMOT) model with application to a PH neutralization process, Proc. of IEEE, 3051-3055, 2003.

[11] A. Gholipour, C. Lucas, B. N. Araabi, and M. Shafiee, Solar activity forecast: spectral analysis and neurofuzzy prediction, Journal of Atmospheric and Solar Terrestrial Physics, 67: 595-603, 2005.

[12] A. Gholipour. C. Lucas, B. N. Araabi, M. Mirmomeni, and M. Shafiee, Extracting the main patterns of natural time series for long-term neurofuzzy prediction, Neural Comput & Applications, doi: 10.1007/s00521-006-0062-x, 2006.

[13] T. A. Johansen, B. A. Foss, NARMAX models using ARMAX models, International Journal of Control, 58(5): 1125-1153, 1993.

[14] O. Nelles, Local linear model tree for on-line identification of time-variant nonlinear dynamic systems, Int. Conf. on Artificial Neural Networks (ICANN), Bochum, Germany, 115-120, July 1996.

[15] M. Mackey, L. Glass, Oscillation and chaos in physiological control systems, Science, 197: 281-287, 1977.

[16] http://homes.esat.kuleuven.be/~smc/daisy.

Combining predictions of a time-series and the first-order difference using bagging of competitive associative nets

Shuichi Kurogi, Shinya Tanaka, and Ryohei Koyama

Department Control Engineering, Kyushu Institute of Technology, Japan

Abstract. This article describes the method which we have used for the prediction competition held at the first ESTSP (European Symposium on Time Series Prediction). In order to obtain small MSE (mean square error) for both 15 and 50 multi-step predictions, we have employed the predictions of the original and the first-order-difference time-series, and combined them. For a good generalization performance on the prediction, we have used the bagging method and competitive associative nets as the base learning predictors.

1 Introduction

The time-series provided for the prediction competition held at the first ESTSP (European Symposium on Time Series Prediction) consists of 875 data points, and the unknown values of the succeeding 50 data points have to be predicted. The performance of the prediction is evaluated by means of MSE_1 and MSE_2 which, respectively, are the MSE (mean square error) obtained for 15 and 50 multi-step predictions next to the provided time-series. In order to reduce both MSE_1 and MSE_2, we make multi-step predictions of the original and the first-order-difference time-series, and combine them.

On the other hand, we use a bagging scheme of the competitive associative net called CAN2 for a good generalization performance in the prediction. Here, the bagging [1, 2] is a method for improving a single predictor, and the CAN2 is a neural net for learning an efficient piecewise linear approximation of a nonlinear function (see [3, 4, 5] for details).

2 Method for the Prediction Competition

Here, we describe our method for the prediction competition after formalizing the prediction problem as follows. Namely, at the competition, the time-series of real values, $y(t)$ $(\in \mathbb{R})$, is given for t in $T^{given} \triangleq \{1, 2, \cdots, t^{given}\}$, where $t^{given} = 875$. A prediction $\widehat{y}(t)$ of $y(t)$ is evaluated by means of $MSE_1 \triangleq MSE(t^{given}, t_1)$ and $MSE_2 \triangleq MSE(t^{given}, t_2)$, where $t_1 = 15$ and $t_2 = 50$, and

$$MSE(t_L, t_p) \triangleq \frac{1}{t_p} \sum_{i=1}^{t_p} \left(y(t_L + i) - \widehat{y}(t_L + i)\right)^2, \tag{1}$$

where t_L represents a point in time for a predictor to learn $y(t)$ for $t \leq t_L$, and t_p represents the prediction horizon.

2.1 Predictions of a time-series and the first-order difference and their combination

Firstly, we suppose that the time-series $y(t)$ $(t = 1, 2, \cdots)$ satisfies

$$y(t) = f(\boldsymbol{x}(t)) + \epsilon(t), \tag{2}$$

where $f(\boldsymbol{x}(t))$ is a nonlinear function of $\boldsymbol{x}(t) \triangleq (y(t-1), y(t-2), \cdots, y(t-k_x))^T$, k_x is the embedding dimension, and $\epsilon(t)$ represents noise. At the competition, we use the bagging of the CAN2 (see below for details) as a learning predictor. The predictor, at first, is used for learning the original time-series $y(t)$ $(t \in T^{given})$ to make the multi-step prediction as

$$\widehat{y}_0(t) = f(\widehat{\boldsymbol{x}}(t), \theta_f), \tag{3}$$

where θ_f represents the parameter values of the predictor, and the elements of the vector $\widehat{\boldsymbol{x}}(t) \triangleq (\widehat{x}(t-1), \widehat{x}(t-2), \cdots, \widehat{x}(t-k_x))^T$ are given by

$$\widehat{x}(t) \triangleq \begin{cases} y(t) & \text{for } t \in T^{given}, \\ \widehat{y}(t) & \text{for } t \in T_p^{pred}. \end{cases} \tag{4}$$

Here, $T_p^{pred} \triangleq \{t_L + 1, t_L + 2, \cdots, t_L + t_p\}$ and $t_p = 50$ for the final prediction.

In order to estimate $MSE_p = MSE(t^{given}, t_p)$ for $p = 1$ and 2, it seems effective to calculate the validation performance given by $MSE(t_L, t_p)$ for several $t_L \leq t^{given} - t_p$, such as $t_L = 825, 800, 725, \cdots$. From several experiments, we found that a good parameter θ_f for MSE_1 is not good for MSE_2. Moreover, the following first-order-difference prediction is found to be effective for reducing MSE_1. Namely, we make the first-order-difference time-series $y_d(t) = y(t) - y(t-1)$, and use a learning predictor to learn $y_d(t)$ for $t = 2, \cdots, t_L$ and get the prediction $\widehat{y_d}(t) = g(\widehat{\boldsymbol{x}_d}(t), \theta_g)$ for $t = t_L + i$ $(i = 1, 2, \cdots, t_1 = 15)$, where $\widehat{\boldsymbol{x}_d}(t) \triangleq (\widehat{x_d}(t-1), \widehat{x_d}(t-2), \cdots, \widehat{x_d}(t-k_{x_d}))^T$ whose elements are generated by means of replacing $x(t)$ and $y(t)$ in Eq.(4) by $x_d(t)$ and $y_d(t)$, respectively. Here, k_{x_d} represents the embedding dimension, $g(\widehat{\boldsymbol{x}_d}(t), \theta_g)$ a nonlinear function of $\widehat{\boldsymbol{x}_d}(t)$, and θ_g represents the parameter values of the predictor. Then, the prediction of $y(t)$ for $t = t_L + 1, t_L + 2, \cdots$ is given by

$$\widehat{y}_1(t) \triangleq y(t_L) + \sum_{j=t_L+1}^{t} \widehat{y}_d(t). \tag{5}$$

Finally, we make the prediction $\widehat{y}(t)$ via combining $\widehat{y}_0(t)$ and $\widehat{y}_1(t)$ as

$$\widehat{y}(t) \triangleq \begin{cases} \widehat{y}_1(t) + \left(\widehat{y}_0(t_L + t_1) - \widehat{y}_1(t_L + t_1)\right)(t - t_L)/t_1 & \text{for } t_L < t < t_L + t_1, \\ \widehat{y}_0(t) & \text{for } t \geq t_L + t_1. \end{cases} \tag{6}$$

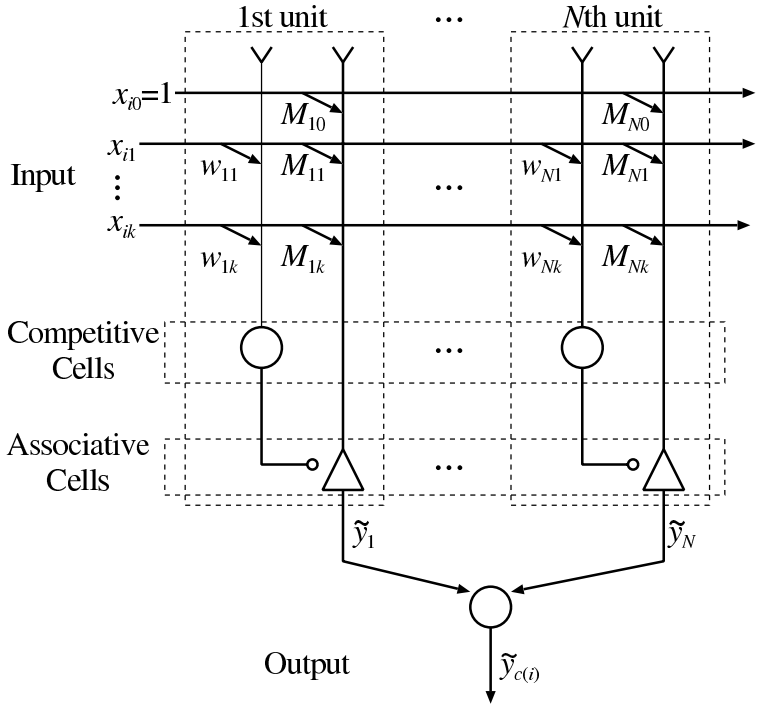

Fig. 1: Schematic diagram of the CAN2

2.2 CAN2 and the bagging for making predictions

2.2.1 Assumptions on the given dataset

Let $D^n \triangleq \{(\boldsymbol{x}_i, y_i) | i \in I^n\}$ be a given training dataset, where $I^n \triangleq \{1, 2, \cdots, n\}$ denotes the index set of the dataset, and $\boldsymbol{x}_i \triangleq (x_{i1}, x_{i2}, \cdots, x_{ik})^T$ and y_i denote an input vector and the target scalar value, respectively. Note that $\boldsymbol{x}_i$ and y_i, respectively, correspond to $\boldsymbol{x}(t)$ and $y(t)$, or $\boldsymbol{x}_d(t)$ and $y_d(t)$, introduced in the previous section. Here, we suppose the relationship given by

$$y_i \triangleq r_i + \epsilon_i = f(\boldsymbol{x}_i) + \epsilon_i, \tag{7}$$

where $r_i \triangleq f(\boldsymbol{x}_i)$ is a nonlinear function of $\boldsymbol{x}_i$, and ϵ_i represents zero-mean noise with the variance σ_i^2.

2.2.2 CAN2

A CAN2 has N units (see Fig. 1). The jth unit has a weight vector $\boldsymbol{w}_j \triangleq (w_{j1}, \cdots, w_{jk})^T \in \mathbb{R}^{k \times 1}$ and an associative matrix (or a row vector) $\boldsymbol{M}_j \triangleq (M_{j0}, M_{j1}, \cdots, M_{jk}) \in \mathbb{R}^{1 \times (k+1)}$ for $j \in I^N \triangleq \{1, 2, \cdots, N\}$. The CAN2 ap-

proximates the above function $f(\boldsymbol{x}_i)$ by

$$\widehat{y}_i \triangleq \widehat{f}(\boldsymbol{x}_i) \triangleq \widetilde{y}_{c(i)} \triangleq \boldsymbol{M}_{c(i)}\widetilde{\boldsymbol{x}}_i, \tag{8}$$

where $\widetilde{\boldsymbol{x}}_i \triangleq (1, \boldsymbol{x}_i^T)^T \in \mathbb{R}^{(k+1)\times 1}$ denotes the (extended) input vector to the CAN2, and $\widetilde{y}_{c(i)} = \boldsymbol{M}_{c(i)}\widetilde{\boldsymbol{x}}$ is the output value of the $c(i)$th unit of the CAN2. The index $c(i)$ indicates the unit who has the weight vector $\boldsymbol{w}_{c(i)}$ closest to the input vector $\boldsymbol{x}_i$, or

$$c(i) \triangleq \underset{j\in I^N}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{w}_j\|. \tag{9}$$

The above function approximation partitions the input space $V = \mathbb{R}^k$ into the Voronoi (or Dirichlet) regions

$$V_j \triangleq \{\boldsymbol{x} \mid j = \underset{i\in I^N}{\operatorname{argmin}} \|\boldsymbol{x} - \boldsymbol{w}_i\|\}, \tag{10}$$

for $j \in I^N$, and performs piecewise linear approximation of the function $f(\boldsymbol{x})$.

Note that the CAN2 has been introduced for utilizing competitive and associative schemes[3, 4, 5], on which there are differences to other similar methods. For example, the method of local linear models [6] uses linear models obtained from K-nearest neighbors of input vectors while the CAN2 utilizes linear models (associative memories) optimized by the learning involving competitive and associative schemes. The CAN2 may be viewed as a mixture-of-experts model that utilizes linear models as experts and competitive scheme as gating. Although the MARS (multivariate adaptive regression splines) model [7] as a mixture-of-experts model executes continuous piecewise linear approximation, the CAN2 executes discontinuous one intending for optimizing each linear model in the corresponding Voronoi region.

Furthermore, we have developed an efficient batch learning method (see [5] for more details), which we also use in the present competition.

2.2.3 Bagging

Let $D_j^{\alpha n*}$ be the jth bootstrap sample set (multiset, or bag) involving αn elements, where the elements in $D_j^{\alpha n*}$ are resampled randomly with replacement from the given training dataset D^n, and $\alpha > 0$. Here, we would like to mention that an element in D^n is not in $D_j^{\alpha n*}$ with the probability $(1 - 1/n)^{\alpha n}$ which approximately is $\exp(-\alpha)$ when n is large. Thus, the number of "individual" or different elements in $D_j^{\alpha n*}$ approximately is $n_{\text{eff}}(\alpha) \triangleq n(1 - \exp(-\alpha))$. For example, $n_{\text{eff}}(1) \simeq 0.632n$ which is used in the conventional bagging methods [1, 2], and $n_{\text{eff}}(0.7) \simeq 0.503n$, which we have employed in the present method (see Section **3** for details).

The bagging (bootstrap aggregation) for estimating the target value $r_i = f(\boldsymbol{x}_i)$ is done by the arithmetic mean given by

$$\hat{y}_i^{b*} \triangleq \frac{1}{b}\sum_{j\in I^b} \hat{y}_i^j, \tag{11}$$

where $\hat{y}_i^j \triangleq \hat{y}^j(\boldsymbol{x}_i)$ denotes the prediction by the jth predictor (CAN2) which has learned the resampled dataset D_j^{can*}.

2.3 Analysis of the present method

Here, we show some analysis of the present method for examining how the method works.

2.3.1 Prediction by means of the first-order difference

As noted before, the prediction via the first-order difference empirically provided a good performance in the 15 step prediction, but not a good performance in the 50 step prediction. One of the reasons for the good performance in the 15 step prediction is supposed to be based on that the range of the input vector $\boldsymbol{x}_d(t) = (y_d(t-1), y_d(t-2), \cdots, y_d(t-k_{x_d}))^T$ to learn the function $y_d(t) = g(\boldsymbol{x}_d(t))$ is smaller than that of of the original input vector $\boldsymbol{x}(t) = (y(t-1), y(t-2), \cdots, y(t-k_x))^T$. Thus, even if there are few training data near the data to be predicted in the original input space, much more training data are available via the first-order difference.

One of the reasons for the bad performance in the 50 step prediction can be understood from the following example; suppose that the given time-series is represented by $y(t) = y(t-1) + a + \epsilon_t$ for $t = 1, 2, \cdots$, where $y(0) = 0$, $a \neq 0$ and ϵ_t represents a noise. Then, the time-series without noise is represented by $r(t) = at$. The first-order difference without noise is written by a linear combination of $y_d(t) = a$ and $y_d(t) = y_d(t-1)$, or $y_d(t) = (1-b)a + by_d(t-1)$ for a certain b $(0 \leq b \leq 1)$. If a predictor learns to predict the ideal first-order difference without noise $\hat{y}_d(t) = (1-b)a + by_d(t-1)$, the prediction of $y(t)$ for $t = t_L + i$ $(i = 1, 2, \cdots)$ is given by $\hat{y}_d(t) = a + (\epsilon_{t_L} - \epsilon_{t_L-1})b^{t-t_L}$. Thus, the reconstructed prediction is given by $y_1(t) = at + \epsilon_{t_L} + (\epsilon_{t_L} - \epsilon_{t_L-1}) \sum_{j=1}^{t-t_L} b^j$. So, the absolute value of the prediction error, $|y_1(t) - r(t)| = |\epsilon_{t_L} + (\epsilon_{t_L} - \epsilon_{t_L-1}) \sum_{j=1}^{t-t_L} b^j|$ increases with the increase of time t for $\epsilon_{t_L} \neq \epsilon_{t_L-1}$ and $b \neq 0$, and the sign of the error becomes positive if $\epsilon_{t_L} > \epsilon_{t_L-1}$ and negative otherwise.

The present method seems to utilize the above advantage and eliminate the above disadvantage of the prediction using the first-order difference.

2.3.2 Coefficients of the bagging

Instead of the bagging prediction given by Eq.(11), a more general aggregation of predictions is given by

$$\tilde{y}_i^{b^*} \triangleq \sum_{j \in I^b} b_j \hat{y}_i^j, \tag{12}$$

where we suppose that $b_j \geq 0$ and $\sum_{j \in I^b} b_j = 1$. Since the prediction $\hat{y}_i^j$ involves the variation caused by the noise ϵ_i in the training data and the variation of the

bootstrap resampling dataset $D_j^{\alpha n*}$ for $j \in I^b$, the mean μ_i, the variation δ_i^j and the variance ρ_i^2 of the prediction $\hat{y}_i^j = \mu_i + \delta_i^j$ are given by

$$\mu_i \triangleq E_{(\epsilon_i, D_j^{\alpha n*})}\left(\hat{y}_i^j\right), \qquad \delta_i^j \triangleq \hat{y}_i^j - \mu_i, \qquad \rho_i^2 \triangleq E_{(\epsilon_i, D_j^{\alpha n*})}\left((\delta_i^j)^2\right) \tag{13}$$

where $E_{(\epsilon_i, D_j^{\alpha n*})}(\cdot)$ is the mean with respect to the variation of ϵ_i and the variation of $D_j^{\alpha n*}$. Since the predictor learns $y_i = r_i + \epsilon_i$, the variation δ_i^j of the prediction is supposed to have a positive correlation with ϵ_i, or

$$E_{(\epsilon_i, D_j^{\alpha n*})}\left(\delta_i^j \epsilon_i\right) > 0. \tag{14}$$

Then, the expectation of the squared prediction error $(\tilde{e}_i^{b^*})^2 = (\tilde{y}_i^{b^*} - y_i)^2$ is given by

$$\begin{aligned} E_{(\epsilon_i, D_j^{\alpha n*})}\left((\tilde{e}_i^{b^*})^2\right) &= \sum_{l \in I^b} b_l^2 \left[(\mu_i - r_i)^2 + E_{(\epsilon_i, D_j^{\alpha n*})}\left((\delta_i^j - \epsilon_i)^2\right)\right], \\ &\geq \frac{1}{b}\left[(\mu_i - r_i)^2 + E_{(\epsilon_i, D_j^{\alpha n*})}\left((\delta_i^j - \epsilon_i)^2\right)\right], \end{aligned} \tag{15}$$

where the inequality is derived by the arithmetic-geometric mean inequality, and the equality holds when b_j is constant. Thus, the mean of the square error of the aggregation takes the minimum when $b_j = 1/b$ ($j \in I^b$). Therefore, the bagging prediction given by Eq.(11) is supposed to be the most effective predictions among the aggregation given by Eq.(12).

2.3.3 Bias and variance decomposition of prediction error

The generalization error or the prediction error for the population data is given by $L^{gen} \triangleq \sum_{i \in I^{pop}} \left(e_i^{b^*}\right)^2$, where I^{pop} indicates the index set of the population, $e_i^{b^*} = \hat{y}_i^{b^*} - y_i$ indicates the prediction error. Let us suppose $\hat{y}_i^{b^*} = \mu_i^{b^*} + \delta_i^{b^*}$, where $\mu_i^{b^*} \triangleq E_{(\epsilon_i, D_j^{\alpha n*})}\left(\hat{y}_i^{b^*}\right) (= \mu_i)$ is the prediction mean, $\delta_i^{b^*} = (1/b)\sum_{j \in I^{b^*}} \delta_i^j$ the variation from the mean. Then, the expectation of L^{gen} is given by

$$E_{(\epsilon_i, D_j^{\alpha n*})}(L^{gen}) = \sum_{i \in I^{pop}} \left((\beta_i^{b^*})^2 + \frac{\rho_i^2}{b} + \sigma_i^2\right), \tag{16}$$

where $\beta_i^{b^*} \triangleq E_{(\epsilon_i, D_j^{\alpha n*})}\left(\hat{y}_i^{b^*} - y_i\right) = \mu_i^{b^*} - r_i$ denotes the bias term, and ρ_i^2/b represents the variance term. Thus, the variance term of the bagging, ρ_i^2/b, can be reduced by the increase of b. Here, we would like to note that in order to reduce the bias term $(\beta_i^{b^*})^2$, we have developed a bagboosting method [8] for the CAN2, but we had few improvements. We think it is because the amount of the bias is not so large. We had a huge computational cost instead of the performance improvement, so we have abandoned the use of the bagboosting method.

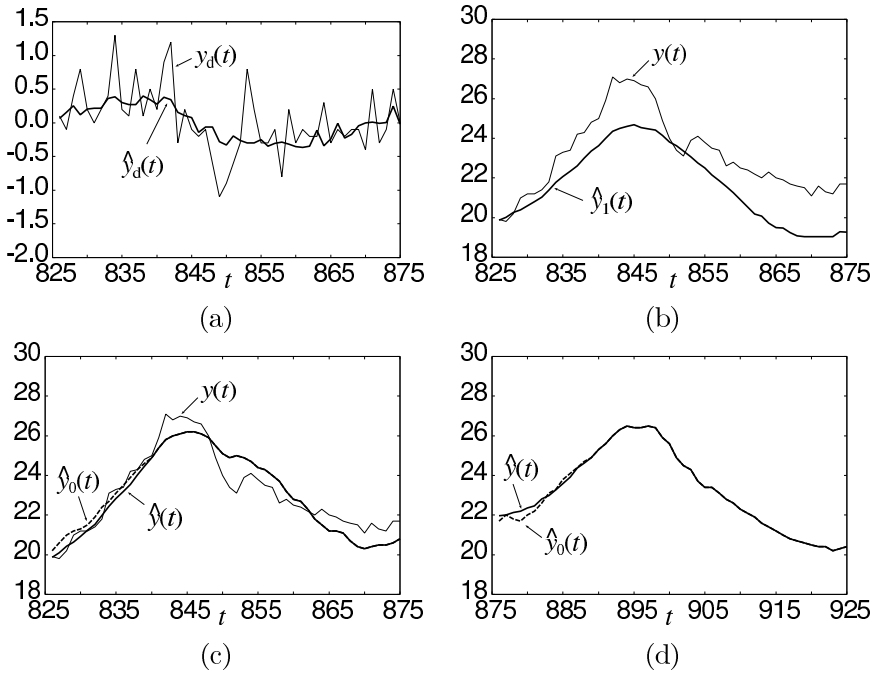

Fig. 2: Prediction results. (a), (b) and (c) show the predictions $\widehat{y}_d(t)$, $\widehat{y}_0(t)$, $\widehat{y}_1(t)$ and $\widehat{y}(t)$ for $t = 826, 827, \cdots, 875$ after learning the training data $y(t)$ for $t = 1, 2, \cdots, 825$, and (d) shows $\widehat{y}_0(t)$ and $\widehat{y}(t)$ for $t = 876, \cdots, 925$ after learning $y(t)$ for $t = 1, 2, \cdots, 875$.

Incidentally, the training (empirical) error is given by $L^{train} \triangleq \sum_{i \in I^n} \left(e_i^{b^*}\right)^2$, and the expectation of L^{train} is given by

$$E_{(\epsilon_i, D_j^{\alpha n *})}(L^{train}) = \sum_{i \in I^n} \left((\beta_i^{b^*})^2 + E_{(\epsilon_i, D_j^{\alpha n *})}((\delta_i^{b^*} - \epsilon_i)^2) \right), \qquad (17)$$

where $\delta_i^{b^*} = (1/b) \sum_{j \in I^{b^*}} \delta_i^j$ or the variation of the prediction has a correlation with ϵ_i as shown in Eq.(14). Thus, we should not select the predictor only from the training error L^{train} because $\delta_i^{b^*}$ may overfit the error ϵ_i.

3 Experimental Results

Here, we show some experimental results. We have examined our method with $y(t)$ $(t = 1, 2, \cdots, 825)$ for training, and $y(t)$ $(t = 826, 827, \cdots, 875)$ for prediction. The result is shown in Fig. 2(a),(b) and (c). From (a), we can see that $\widehat{y}_d(t)$ seems to predict $y_d(t)$ very well although the magnitude of the fluctuation of $\widehat{y}_d(t)$ is not so big as $y_d(t)$. From (b), we can see that $\widehat{y}_1(t)$ is near $y(t)$ at the begging of the prediction, but the prediction error seems to increase with

the increase of time. From (c), the prediction $\widehat{y}_0(t)$ seems to be better than the prediction $\widehat{y}_1(t)$ shown in (b) over the prediction period. Furthermore, the combined prediction $\widehat{y}(t)$ is slightly better than $\widehat{y}_0(t)$. Precisely, the MSE of $\widehat{y}_0(t)$ and $\widehat{y}(t)$ was $MSE(825, 15) = 0.16467$ and 0.1142, respectively, for the 15 step prediction, and $MSE(825, 50) = 0.5598$ and 0.5446, respectively, for the 50 step prediction.

The above results are obtained with $k_x = 220$ and $N_f = 40$, where k_x is the embedding dimension of $\widehat{\boldsymbol{x}}(t)$ for $\widehat{y}(t) = f(\widehat{\boldsymbol{x}}(t), \theta_f)$, and N_f represents the number of units of the CAN2 as one of the parameters represented by θ_f. The number of training data given by $(\boldsymbol{x}(t), y(t))$ is $n = t^{given} - k_x - 1 = 654$, and the single CAN2 has a possibility to learn $k_x \times N_f = 8800$ linearly independent data, so that the single CAN2 is able to learn all of the 654 training data almost perfectly (the MSE for the training data was about 1.0×10^{-14}), and it could reproduced the time-series $y(t)$ from the $t = k_x + 1 = 221$ to 875 by means of the multi-step prediction after learning all training data. However, as explained in Section **2.3.3**, we shoud not use this single CAN2 as the predictor only from the point of view of its small training error. We did use the bagging of the CAN2 from the point of view of the performance for the validation test as explained above, such as $MSE(825, 15)$ and $MSE(825, 50)$, where the bagging provided some variation to the prediction error owing to the bootstrap resampling scheme, and it provided small MSEs for the validation dataset.

Furthermore, we would like to note that we have used $\alpha = 0.7$ and $b = 20$ for the bagging because they provided good generalization performance in many validation tests. Although $\alpha = 0.7$ indicates that only the half of the training data is used for training each base predictor (see Section **2.2.3**), this may have provided a good variation of the prediction error to each base predictor for the bagging prediction to achieve good generalization performance.

4 Conclusion

We have described our method used for the prediction competition at the first ESTSP. The method employs the predictions of the original and the first-order difference time-series in order to reduce the errors for both the 15 and 50 multi-step predictions. The method also uses a version of bagging for a good generalization performance.

This work was partially supported by the Grant-in-Aid for Scientific Research (B) 16300070 of the Japanese Ministry of Education, Science, Sports and Culture.

References

[1] L. Breiman, Bagging predictors, *Machine Learning*, 24:123-140, 1996.

[2] J.G. Carney and Padráig Cunningham, The NeuralBAG algorithm: Optimizing generalization performance in bagged neural networks, *Proceedings of ESANN'1999*, 21-23, 1999.

[3] S.Kurogi, Asymptotic optimality of competitive associative nets for their learning in function approximation, *Proceedings ICONIP2002*, 507-511, 2002.

[4] S.Kurogi, "Asymptotic optimality of competitive associative nets and its application to incremental learning of nonlinear functions," IEICE D-II (in Japanese), J86-D-II-2, 184/194 (2003)

[5] S. Kurogi, T. Ueno and M. Sawa: A batch learning method for competitive associative net and its application to function approximation, Proc. of SCI2004, no.V, 24–28 (2004)

[6] J.D.Farmer and J.J.Sidorowich: Predicting chaotic time series. Phys. Rev.Lett., 59:845–848 (1987)

[7] J.H.Friedman: Multivariate adaptive regression splines. Ann Stat, 19, 1-50 (1991)

[8] M.Dettling, Bagboosting for tumor classification with gene expression data, *Bioinformatics*, 20(18)3583-3593, 2004

Introducing a new learning method for fuzzy descriptor models with the application to predict solar activity

Masoud Mirmomeni[1], Masoud Shafiee[2] and Caro Lucas[3]

1- Control and Intelligent Processing Center of Excellence, Electrical and Computer Eng. Department, University of Tehran, Tehran, Iran
2- Amirkabir University of Technology, Electrical Eng. Department, Tehran, Iran
3- School of Cognitive Sciences, Institute for studies in theoretical Physics and Mathematics, Tehran, Iran

m.mirmomeni@ece.ut.ac.ir, mshafiee@aut.ac.ir, lucas@ipm.ir

Abstract. The cyclic solar activity has significant effect on earth, climate, satellites and space missions. Several methods have been introduced for the prediction of sunspot number, which is a common measure of solar activity. In the past two decades, descriptor systems and related fuzzy descriptor models have been the subjects of interest due to their many practical applications in modeling complex phenomena. This paper introduces a new learning method, Generalized Locally Linear Model Tree (GLoLiMoT) algorithm as an intuitive incremental learning algorithm to tune the parameters of fuzzy descriptor models to predict two solar activity indexes: sunspot number and DST index. Simulation results depict the power of this method in predicting such chaotic dynamics in comparison with other methods.

Keywords. Chaotic dynamics, solar activity, fuzzy descriptor, neurofuzzy, GLoLiMoT.

1 Introduction

Among the various conditions that affect space weather, the sun-driven phenomena dominate the others. Origin of many space weather changes is the solar activity which varies in an eleven year period, called solar cycle. The solar cycle consists of a period of activity, the solar maximum, and a period of quiet, the solar minimum. During the years of solar maximum there are more solar flares causing significant increase in solar cosmic ray intensity. The high energy particles disturb communication systems and affect the lifetime of satellites. Coronal mass ejections and solar flares are the origin of shocks in solar wind and cause geomagnetic disturbances in earth's magnetosphere. High rate of geomagnetic storms and sub-storms result in atmosphere heating and drag of low earth orbit (LEO) satellites.

In the past decade, several methods have been proposed to predict the solar activity in advance [1-4]. Following the achievements in the field of chaotic systems, several methods can be used in the prediction of solar activity indexes; namely polynomial function approximation, reconstructions using Lyapunov exponents,

detecting periodicity in chaotic time series and the intelligent predictions based on neural networks and neurofuzzy models [5].

On the other hand, descriptor systems and related fuzzy descriptor models have been the subjects of interest over the last two decades due to their many practical applications in describing complex systems [6]. Such systems describe a wider class of systems, including physical models and non-dynamic constraints. It is well known that descriptor system is much tighter than the state-space expression for representing real independent parametric perturbations [7]. This motivates one to use the fuzzy descriptor models in the prediction of some solar activity indexes which have complex dynamics.

This paper proposes a new method, which is the extension of Locally Linear Model Tree algorithm (LoLiMoT) to fuzzy descriptor models [8] for adjusting the parameters of such systems for prediction. This method is called the Generalized Locally Linear Model Tree algorithm (GLoLiMoT). To show the advantage of this method, the performance of fuzzy descriptor model with this extended learning algorithm is compared with several neural and neurofuzzy models in the prediction of two solar activity indexes: sunspot number and disturbance storm time (DST) index. Results depict the great performance of this method in prediction of solar activity compare to the other neurofuzzy models.

The remaining of this paper is structured as follows. Section 2 briefly introduces nonlinear descriptor system and presents its characteristics compare to the regular systems and address some complex phenomena which can be used for some good test beds to show the performance of such systems in describing the characteristics of such phenomena. Section 3 introduces fuzzy descriptor models and discuss about their characteristics. Section 4 is devoted to describe the learning methodology which is used for fuzzy descriptor models to predict solar activity. In section 5, the fuzzy descriptor model is used to predict sunspot number and DST index to show the performance of fuzzy descriptor model in comparison with other methods. The last section contains the concluding remarks.

2 Descriptor systems

A singular implicit differential equation is an implicit ordinary differential equation which takes the form of

$$F\left(\dot{x}(t), x(t), u(t), t\right) = 0, \qquad \dot{x}(t_0) = x_0 \tag{1}$$

where x is an n-dimensional state vector, u is an m dimensional control vector, t is time and the Jacobian matrix $\dfrac{\partial F}{\partial \dot{x}}$ is singular. Such implicit systems often arise in a wide variety of cases, e.g. in electrical power system, flight control, robotics, microeconomic systems, and contact or constrained problems in mechanic. *A* system which is described by singular implicit differential equation is called a singular system. Singular systems are often referred to as differential algebraic equations because they frequently are a mixture of differential and algebraic equations, that is, they take the form of

$$\dot{x}(t) = f\left(x,u,t\right)$$
$$0 = g\left(x,u,t\right) \tag{2}$$

One can define a matrix E such that

$$E\dot{x}(t) = F\left(x,u,t\right),$$
$$E = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}, \tag{3}$$
$$F\left(x,u,t\right) = \begin{bmatrix} f\left(x,u,t\right) \\ g\left(x,u,t\right) \end{bmatrix}$$

This general form of differential equation is used as a canonical form of nonlinear singular systems in many papers. In addition these systems are also called as descriptor systems because they are the way in which the system is initially described. Other names for descriptor systems are constrained systems, degenerate systems, generalized state-space systems, semi-state systems, non-canonic, and differential equation on a manifold [9]. These systems arise in the study of robotics, optimal control, economics, large-scale interconnected systems, etc [10, 11].

It can be seen that if E is regular, the *implicit* ODE (3) is equivalent to the *explicit* ODE $\dot{x}(\mathrm{t}) = E^{-1}F\left(x(t),u(\mathrm{t})\right)$. This case has been quite thoroughly studied and now rather well understood. When E is singular in (3), resulting in what we shall term a *generalized state-space system* or *descriptor system*, this behaviour is considerably modified. In contrast to *regular state-space system* we find that the number of degrees of freedom of the system, i.e., the number of independent values that $Ex(0.)$ can take, is now evidently reduced to

$$\mathrm{f} \triangleq \operatorname{rank} E < n \tag{4}$$

It is proposed the term *generalized order* for f [12]. Therefore, in such systems state spaces have to satisfy some constraints and state variables have to be on a manifold in state space. In addition, it is proved that the output of such systems may include some impulsive motions even if there is no impulse input to the system [12]. Such characteristics show the power of descriptor systems in describing complex phenomena. For example, in modelling sub-storm dynamics of magnetosphere which its prediction is considered as one of the case studies in this paper, it is proved that both the surface and the corresponding circulation flows turn out to be surprisingly close to a very simple low-dimensional scheme of the magnetospheric sub-storm as a cusp catastrophe (inverse bifurcation) first proposed by Lewis [13] and illustrated in Fig. 1 where z Parameter is the state parameter and c_1 and c_2 are the control parameters [13].

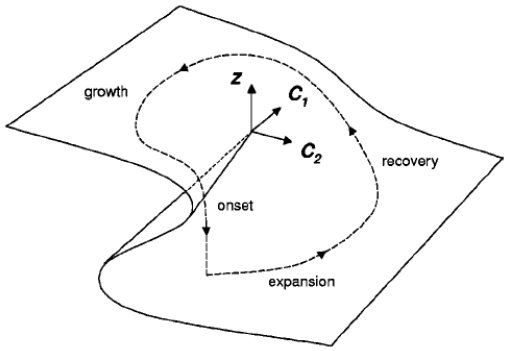

Fig. 1: Hypothetical cusp catastrophe manifold that was expected to approximate the sub-storm dynamics of the magnetosphere according to the model [13]. The evolution of an isolated sub-storm is shown by dashed arrows.

3 Fuzzy descriptor models

In this section the mathematical formulation of fuzzy descriptor models are considered [8]. The fundamental approach with such systems is dividing the input space into small linear subspace with fuzzy validity functions and their appropriate linear descriptor systems. Therefore the total fuzzy descriptor model is an extended neuro-fuzzy network with one hidden layer for locally linear descriptor systems and a linear neuron in the output layer which simply calculates the weighted sum of the outputs of locally linear descriptor systems. Therefore, in this paper a fuzzy descriptor model can be defined by extending the T-S fuzzy model. The fuzzy descriptor model is defined as

Rule *i:* If $z_1(t)$ is M_{1i} and … and $z_n(t)$ is M_{ni}

$$\text{Then: } \begin{cases} E_i \dot{x}(t) = A_i x(t) + B_i u(t) \\ y(t) = C_i x(t) \end{cases} \tag{5}$$

where $x(t) \in R^n$, $y \in R^p$, $u(t) \in R^m$. M_{ij}, is the fuzzy set and r is the number of if-then rules. $x(t) \in R^n$ is the state vector, $u(t) \in R^m$ is the input vector, $y \in R^p$ is the output vector, $E_k \in R^{n \times n}$, $A_i \in R^{n \times n}$, $B_i \in R^{n \times m}$, and $C_i \in R^{p \times n}$. $z_1(t) \sim z_n(t)$ are the premise variables. Each linear state and output equations in the consequent parts are called "descriptor subsystem".

The overall fuzzy model is achieved by fuzzy 'blending' of the linear descriptor subsystems. Given a pair of $(x(t),u(t))$, the final output of the fuzzy system is inferred as follows:

$$\begin{aligned} \sum_{i=1}^{r} h_i(z(t)) E_i \dot{x}(t) &= \sum_{i=1}^{r} h_i(z(t)) \left(A_i x(t) + B_i u(t) \right) \\ y(t) &= \sum_{i=1}^{r} h_i(z(t)) C_i x(t) \end{aligned} \tag{6}$$

where $\omega_i(z(t)) = \prod_{j=1}^{n} M_{ji}(z_j(t))$. In (6), $h_i(z(t)) = \omega_i(z(t)) \Big/ \sum_{i=1}^{r} \omega_i(z(t))$. $h_i(z(t))$ can be regarded as the normalized weight of each if-then rule. $M_{ji}(z_j(t))$ is the membership if $z_j(t)$ in M_{ji}. Defining $x^*(t) = \lfloor x^T(t) \quad \dot{x}^T(t) \rfloor^T$, the fuzzy descriptor system (6) can be written as

$$\begin{aligned} E^* \dot{x}^*(t) &= \sum_{i=1}^{r} h_i(z(t)) \left(A_i^* x^*(t) + B_i^* u(t) \right) \\ y(t) &= \sum_{i=1}^{r} h_i(z(t)) C_i^* x^*(t) \end{aligned} \tag{7}$$

where

$$E^* = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}, \qquad A_{ik}^* = \begin{bmatrix} 0 & I \\ A_i & -E_k \end{bmatrix}$$

$$B_i^* = \begin{bmatrix} 0 \\ B_i \end{bmatrix}, \qquad C_i^* = \begin{bmatrix} C_i & 0 \end{bmatrix}$$

In this paper, the validity functions are chosen as normalized Gaussians. Each Gaussian validity function has two parameters, center c_{ij} and standard deviation σ_{ij}. There are *M.m* parameters for nonlinear hidden layer. In addition, in this paper fuzzy descriptor model is used to predict solar activity as natural chaotic dynamics. Therefore it is obvious that in this application the dynamic system has not control input $u(t)$. Therefore, fuzzy descriptor model for prediction application can be written as

$$\begin{aligned} E^* \dot{x}^*(t) &= \sum_{i=1}^{r} h_i(z(t)) \left(A_i^* x^*(t) \right) \\ y(t) &= \sum_{i=1}^{r} h_i(z(t)) C_i^* x^*(t) \end{aligned} \tag{8}$$

4 Learning methodologies

This section is devoted to describe new learning method for fuzzy descriptor models to adjust its two kinds of parameters. As it said before, the consequent part of a fuzzy descriptor model, is a linear descriptor subsystem which is an improper system by its own. Therefore, to adjust parameters of the consequent parts it is need to use an identification method which is proper for improper systems. Unfortunately, there was no such a well-known method to identify the parameters of a descriptor system. Therefore, in this paper, first of all it is tried to develop a method which could identify the parameters of a descriptor systems. In following subsection this new method is described.

4.1 A quasi-static algorithm for system identification of linear descriptor systems

Consider a linear descriptor system such as

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{9}$$

The output and input under zero initial conditions (i.e., Ex(0_)=0*)* are related by the transfer function G(s), as follows:

$$G(s) = C(sE - A)^{-1}B \tag{10}$$

It is found that the transfer function $G(s)$ may no longer be strictly proper, in which case it may be written as the sum of a strictly proper part $\bar{G}(s)$ and a polynomial part $D(s)$. Therefore, we have:

$$G(s) = \bar{G}(s) + D(s) \tag{11}$$

where

$$\bar{G}(s) = \bar{C}\left(sI - \bar{A}\right)^{-1}\bar{B}, \qquad \text{strictly proper} \tag{12a}$$

and

$$D(s) = \tilde{C}\left(I - s\tilde{E}\right)^{-1}\tilde{B} = \tilde{C}\left(I + s\tilde{E} + \cdots + s^{v}\tilde{E}^{v}\right)\tilde{B}, \quad \text{polynomial} \tag{12b}$$

$\bar{A}$, $\bar{B}$, $\bar{C}$, $\tilde{B}$, $\tilde{C}$ and $\tilde{E}$ are came from a restricted standard equivalence of the system (9). Here v is less than the size of $\tilde{E}$, since $\tilde{E}$ is *nilpotent* (i.e. has all eigenvalues = 0) [12].

It is obvious that the polynomial subsystem in discrete domain will be a moving average subsystem. Fortunately, each sub system could be identified by classical identification methods. Therefore, one can adjust the parameters of a descriptor system by decoupling it in to two subsystems, and then adjust these parameters simultaneously. The quasi-static algorithm to identify parameters of linear descriptor system is as follows:

- To identify the parameters of the strictly proper subsystem, consider the output of the polynomial part (which is not identified yet) as a measurement noise to the strictly proper part.
- Estimate an ARX model $A(q)y(k) = B(q)u(k) + y_2(k)$ from the data $\left\{\underline{u}(k), \underline{y}(k)\right\}$ by

$$\hat{\theta}_{ARX} = \left(\underline{X}^{T}\underline{X}\right)^{-1}\underline{X}^{T}\underline{y} \tag{13}$$

- Calculate the prediction error of this ARX model

$$e_{ARX}(k) = \hat{A}(q)y(k) - \hat{B}(q)u(k) \tag{14}$$

whose $\hat{A}(q)$ and $\hat{B}(q)$ are determined by $\hat{\theta}_{ARX}$.

- To identify the parameters of the polynomial part, consider the output of the strictly proper part (which is identified in this iteration) as a measurement noise to the polynomial part.
- Estimate the d_i parameters of the following FIR model by least squares

$$e_{ARX}(k) = D(q)u(k) \tag{15}$$

This algorithm can be iterated until the convergence is reached. This method yields the linear descriptor systems' discrete transfer function in frequency domain. This model has to be converted to the state space model to be useful for fuzzy descriptor models. This could be done by Silverman-Ho algorithm which gets the state space model of the descriptor system via its transfer function [14, 15].

4.2 An incremental learning algorithm for premise part of the fuzzy descriptor models

After adjusting linear descriptor system's parameters, it is time to adjust the parameters of validity functions for each locally linear descriptor system. In this section GLoLiMoT algorithm (Generalized Locally Linear Model Tree algorithm) which is based on LoLiMoT algorithm for locally linear neuro-fuzzy models [16] is introduced to adjust both validity functions' parameters and locally linear descriptor systems' parameters. The GLoLiMoT algorithm is described in five steps:

1. *Start with an initial model:* start with a single LLDM (Locally Linear Descriptor Model), which is a global linear model over the whole input space with $h_1(\underline{z}) = 1$ and set $M = 1$. If there is a priori input space partitioning it can be used as the initial structure.
2. *Find the worst LLDM*: Calculate a local loss function e.g. MSE (Mean Squared error) for each of the $i = 1, \ldots, M$ LLDMs, and find the worst performing LLDM.
3. *Check all divisions*: The worst LLDM is considered for further refinement. The hyper rectangle of this LLDM is split into two halves with an axis orthogonal split. Divisions in all dimensions are tried, and for each of p divisions, following steps are carried out:
   a. Construction of the multi-dimensional membership functions for both generated hyper rectangles.
   b. Construction of all validity functions.
   c. System identification of linear descriptor systems for both generated hyper rectangles by decoupling method introduced in former subsection.
   d. Construction of new fuzzy descriptor system according to new linear descriptor systems in state space form which is produced by Silverman-Ho algorithm.
   e. Calculations of the loss function for current overall model.
4. *Find the best division*: The best of the p alternatives checked in step 3 is selected, and the related validity functions and LLDMs are constructed. The number of LLDM neurons is incremented to $M = M + 1$.
5. *Test the termination condition*: If the termination condition is met, then stop, else go to step 2.

5 Solar activity forecasting

5.1 Sunspot number forecasting

The sunspot number is a good measure of solar activity and is computed according to the Wolf formulation:

$$R=k\,(10g+s) \qquad (16)$$

The GLoLiMoT algorithm is implemented as a MATLAB m-file and is used to predict the sunspot number. The number of iterations is also optimized by an intelligent program: the model will be checked by the test data in each iteration and the training will be stopped when the mean square error (MSE) of validation data starts to increase. In this way, the over-fitness is avoided and the most accurate prediction is prepared. Three other networks have been implemented to be compared with fuzzy descriptor models and its algorithm GLoLiMoT; the MLP network with conjugate gradient learning method, the RBF network, and its predecessor LLM network with LoLiMoT learning method. All of these models are compared in their optimum performance. Table 1 contains the results of several methods; the RBF and the MLP (Multi Layered Perceptron) and the LLM (Locally Linear Model) networks. Fig. 3 presents the prediction of sunspot number (total test set) by GLOLIMOT algorithm. This algorithm shows good performance in the solar maximum (peak points of sunspot number), especially in 1958, while the other methods do not.

	Test set 1	Test set 2	Test set 3	Total test
MLP	0.2	0.3	0.22	0.23
RBF	0.1186	0.184	0.1421	0.1392
LoLiMoT	0.0702	0.1518	0.1519	0.1136
GLoLiMoT	0.0634	0.1056	0.1392	0.0873

Table 1. *NMSE* of several methods in the prediction of yearly sunspot numbers

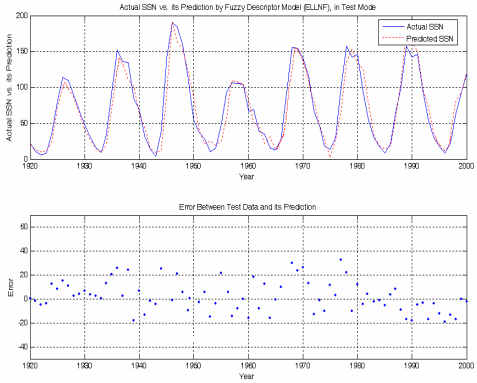

Fig 3: prediction of sunspot number by a fuzzy descriptor model with GLoLiMoT algorithm; Upper: Predicted and Observed values of test set, Lower: Prediction error.

5.2 DST index forecasting

In this section, it is tried to predict the daily average of the DST index during 1957 to 2005. 11789 data (from 1957 to 1990) is used to adjust the fuzzy descriptor parameters and about 5800 (from 1990 to 2005) data is kept to test the performance of the fuzzy descriptor system in predicting daily average of the DST index. Figure 4 depicts the performance of the fuzzy descriptor model in predicting the daily average of the DST index in test mode. It can be easily seen that the fuzzy descriptor system in this problem has very good results.

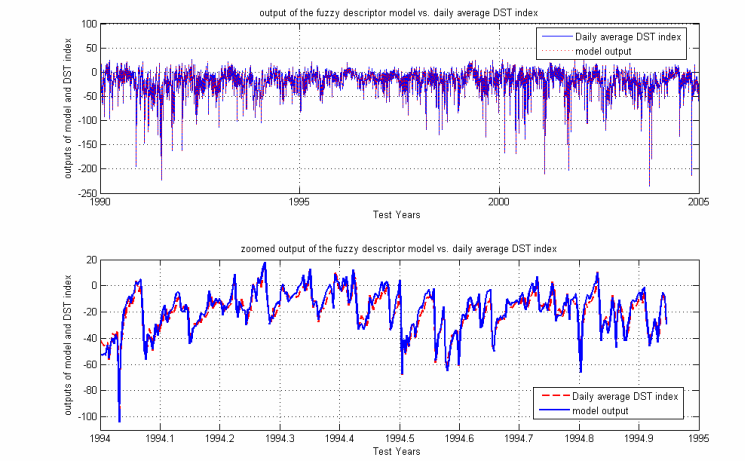

Fig 10: One step ahead prediction of daily average of the DST index by a fuzzy descriptor model; Upper: Predicted and Observed values of test set, Lower: Predicted and Observed values of test set during 1994.

The comparison between this combined method results and MLP technique's result in daily average predicting DST from 1957 to 2005 is presented in Table 2.

Method	NMSE in predicting DST index
MLP	0.0952
GLoLiMoT	0.0384

Table 2. NMSE of proposed method in compare of the result of MLP method in the prediction of daily average DST index.

6 Discussion and Conclusion

This paper has defined a fuzzy descriptor system by extending the ordinary T-S fuzzy model. Such system can be used as predictor when it is trained by constructing learning methods. In this research, several optimization methods have been used with GLoLiMoT algorithm, to predict two important measures of solar activity: the sunspot number and DST index. By optimizing the number of neurons, the splitting

ratio and the standard deviations, an accurate prediction has been provided. The especially low prediction error of the proposed method in the peak points of sunspot number (solar maximum) even with few neurons in its structure is an interesting achievement. Due to its high generalization and low prediction error, this method can be used in predicting the solar activity several years in advance.

References

[1] G. M. Brown, The peak of solar cycle 22: prediction in retrospect, *Annales Geophysicae*, 10:453-470, 1992.

[2] R. Thompson, A technique for predicting the amplitude of solar cycle, *Solar Physics*, 148: 383, 1993.

[3] A. Gholipour, A. Abbaspour, , B. N. Araabi, and C. Lucas, Enhancements in the prediction of solar activity by locally linear model tree, proceeding of *MIC2003: 22nd Int. Conf. on Modeling, Identification and Control*, Innsbruck, Austria, 158-161, 2003.

[4] A. Gholipour, C. Lucas, B. N. Arrabi, and M. Shafiee, Solar activity forecast: spectral analysis and neurofuzzy prediction, Journal of Atmospheric and Solar Terrestrial Physics, 67: 595-603, 2005.

[5] C. Lucas, A. Abbaspour, A. Gholipour, B. N. Araabi, and M. Fatourechi, Enhancing the performance of neurofuzzy predictors by emotional learning algorithm, *Informatica*, 27(2): 165-174, 2003.

[6] D. G. Luenberger, Dynamic equation in descriptor forms, IEEE Tran. on Auto. Cont., AC-22: 312-321, 1977.

[7] G. Lu, D. W. C. Ho, Generalized quadratic stabilization for perturbated discrete-time singular systems with delayed state, 4th Int. Conf. on Control and Automation, Montreal, Canada, June 10-12, 2003.

[8] T. Taniguchi, K. Tanaka, K. Yamafuji, and H. O. Wang, Fuzzy Descriptor Systems: Stability Analysis and Design via LMIs, Proceeding of the American Control Conference, San Diego, CA, June 1999.

[9] L. Xiaoping, Solvability of nonlinear singular systems-part II: the case with inputs, Proceeding of the American Control Conference, Seattle, Washington, June 1995.

[10] M. Shafiee, S. Amani, Optimal control for a class of singular systems using neural network, Iranian Journal of Science and Technology, Transaction B, engineering, 29(B1), 2005.

[11] Y. Wang, Z. Q. Sun, and F. C. Sun, Robust fuzzy control of a class of nonlinear descriptor systems with time-varying delay, Int. Journal of Control, Automatics and Systems, 2(1): 76-82, Mar. 2004.

[12] G. C. Verghese, B. C. Levy, and T. Kailath, A generalized state-space for singular systems, IEEE Tran. on Automatic Control, AC-26(4), August 1981.

[13] M. I. Stinov, A. S. Sharma, and K. Papadopoulos, Modeling sub-storm dynamics of the magnetosphere: from self-organization and self-organized criticality to non-equilibrium phase transition, Physical Review E, 65(016116): 1-11, 2001.

[14] L, Dai, *Singular Control Systems*, Springer, New York, 1989.

[15] J, Wang, Q. Zhang, W. Liu, X. Xin, and V. Sreeram, H^{∞} model reduction for singular systems, Proceeding of 2004 American Control Conference, Boston, Massachusetts, pp. 119-124, June 30-July 2, 2004.

[16] O. Nelles, *Nonlinear System Identification*, Springer, Berlin, 2001.

Estimating the Number of Components of a Mixture Autoregressive Model

M. Olteanu and J. Rynkiewicz

SAMOS-MATISSE, CES, Universite Pantheon-Sorbonne
90 Rue de Tolbiac, 75013 Paris, France

Abstract. In this paper we are interested in estimating the number of components of a mixture autoregressive (MAR) model. Usually, when estimating the parameters of such a model, a fixed number of components is considered "a priori", although the general case of an unknown number should be more interesting to study. However, not fixing the number of components leads to non-identifiability problems which complicate the task. Recently, the consistence of a penalized marginal-likelihood criterion for mixture models and hidden Markov models was proven by Keribin (2000) and, respectively, Gassiat (2002). We extend their method to mixtures of autoregressive models for which a penalized-likelihood criterion is proposed. We prove the consistency of the estimate under some hypothesis which involve essentially the bracketing entropy of the generalized score-functions class and we verify these hypothesis in the Gaussian case by reparameterizing the model in order to avoid non-identifiability problems. Some numerical examples illustrate the result and its convergence properties.

1 Introduction

Although linear models have been the standard tool for time series analysis for a long time, their limitations have been underlined during the past twenty years. Real data often exhibit characteristics that are not taken into account by linear models. Financial series, for instance, alternate strong and weak volatility periods, economic series are often related to the business cycle and thus switch from recession to normal periods, while the series of river flows usually have heavy tails. Several solutions were proposed to overcome these problems. Without looking for exhaustivity, let us remind some of the most popular at the moment : conditional heteroscedatic models like ARCH, GARCH (Engle, 1982, Bollerslev, 1986) and their generalizations, threshold and piecewise linear models (Tong, 1983), multilayer perceptrons and autoregressive switching Markov models (Hamilton, 1989).

In this paper, we study a particular class of regime switching models which are mixture autoregressive (MAR) models. Although they can be regarded as autoregressive switching Markov models with transition probabilities independent of the previous states of the hidden Markov chain, their formal definition was introduced by Wong and Li (2000). Their main justification for this class of models was the necessity of introducing a time series model which accounts

for multimodal conditional distributions and also captures the conditional heteroscedasticity. In their paper, they gave sufficient conditions for weak stationarity and proposed an EM algorithm to estimate the parameters. However, the problem of model selection via some information criterion like AIC or BIC is studied only empirically by simulations results, while the theoretical justification of the consistence was left as an open problem.

Indeed, estimating the number of components for a mixture model or, more generally, for an autoregressive switching Markov model is a complex problem which arises from the non-identifiability of the parameters. In these cases, the Fisher information matrix is degenerate, the usual regularity conditions do not hold and the classical theory for the convergence of the likelihood ratio test statistic does not apply.

However, several ideas and methods were proposed to estimate the number of components in the particular case of mixture models (constant regression functions) : various non-parametric techniques in Henna (1985), Roeder (1994) or Izenman and Sommer (1998), moment techniques in Lindsay (1983) or Dacunha-Castelle and Gassiat (1997) and penalized maximum-likelihood in Leroux(1992), Keribin (2000) and Gassiat (2002). In this paper, we extend the result of Gassiat (2002) to MAR models and prove the consistence of the BIC criterion using some empirical processes techniques.

The rest of the paper is organized as follows : in Section 2 we recall the definition of MAR models and state sufficient conditions for strict stationarity and ergodicity. Afterwards, we introduce the penalized likelihood estimate for the number of components and state the result of consistence. Section 3 is concerned with verifying the hypothesis of the main result in the Gaussian case, while Section 4 provides some simulation results which illustrate the stability and the speed of convergence of the BIC criterion. Some open questions, as well as some possible extensions are discussed in the conclusion.

2 Penalized likelihood estimate for the number of components of a MAR model

Throughout the paper, we shall consider that the number of lags is known and, for ease of writing, we shall set the number of lags at one, the extension to l time-lags being immediate.

2.1 The model - stationarity and ergodicity conditions

Let us consider the real-valued time series Y_t which verifies the true model

$$(1) \qquad Y_t = F^0_{X_t}(Y_{t-1}) + \varepsilon_{X_t}(t)$$

where

- X_t is a sequence of i.i.d. variables with values in a finite space $\{1, ..., p_0\}$ and probability distribution π^0
- for every $i \in \{1, ..., p_0\}$, $F^0_i(y) = a^0_i y + b^0_i$, $y \in \mathbb{R}$ is a linear function

- for every $i \in \{1,...,p_0\}$, $\varepsilon_i(t)$ is an i.i.d. noise with density f_i^0 strictly positive with respect to the Lebesgue measure and depending on the parameter θ_i^0. The global parameter of the model is then $\left(p_0, (\pi_i^0, a_i^0, b_i^0, \theta_i^0)_{i=1,...,p_0}\right)$.

The main convergence result will be proven for a stationary and ergodic process. Sufficient conditions for strict stationarity and ergodicity were given by Yao and Attali (2000) in the general case of autoregressive switching Markov models. Their result may be adapted easily to MAR models by replacing the invariant distribution of the hidden Markov chain with the probability distribution of X_t.

Thus, under the hypothesis

$$\textbf{(HS)} \quad (\exists)\ s \geq 1 \text{ such that } E|\varepsilon_1|^s < \infty \text{ and } \sum_{i=1}^{p_0} \pi_i^0 \left|a_i^0\right|^s < 1$$

the model (1) has a unique strictly-stationary solution Y_t, geometrically-ergodic and with invariant probability measure admiting s-order moments. Let us remark that hypothesis **(HS)** does not request every component to be stationary and that it allows non-stationary "regimes" as long as they do not apper too often. On the other hand, let us also note that **(HS)** is immediately verified if every component is stationary, that is $\left|a_i^0\right| < 1$ for every $i \in \{1,...,p_0\}$.

2.2 Construction of the penalized likelihood estimate

Let us consider an observed sample $\{y_1,...,y_n\}$ of the time series Y_k. Then, for every observation y_k, the conditional density with respect to the previous y_{k-1} and marginally in X_k is

$$f(y_k \mid y_{k-1}) = \sum_{i=1}^{p_0} \pi_i^0 f_i^0 \left(y_k - F_i^0(y_{k-1})\right)$$

As the goal is to estimate p_0, the number of components of the model, let us consider all possible conditional densities up to a maximal number of regimes P, a fixed positive integer. We shall consider the class of functions

$$\mathcal{G}_P = \bigcup_{p=1}^{P} \mathcal{G}_p,$$

$$\mathcal{G}_p = \left\{ g \mid g(y_1,y_2) = \sum_{i=1}^{p} \pi_i f_i(y_2 - F_i(y_1)),\ \pi_i \geq 0,\ \sum_{i=1}^{p} \pi_i = 1 \right\}$$

where, for all $i = 1,...,p$, $F_i(y) = a_i y + b_i$ and f_i is a strictly positive density with respect to the Lebesgue measure depending on θ_i.

(HC) We shall assume throughout the following that the parameters $\{(\pi_i, a_i, b_i, \theta_i), i = 1,...,p\}$ belong to a compact set.

For every $g \in \mathcal{G}_P$ we define the number of components as

$$p(g) = min\{p \in \{1,...,P\},\ g \in \mathcal{G}_p\}$$

and let $p_0 = p(f)$ be the true number of regimes.

We can now define the estimate $\hat{p}$ as the argument $p \in \{1, ..., P\}$ maximizing the penalized criterion

$$(2) \qquad T_n(p) = sup_{g \in \mathcal{G}_p} l_n(g) - a_n(p)$$

where $l_n(g) = \sum_{k=2}^{n} log\, g(y_{k-1}, y_k)$ is the log-likelihood marginal in X_k and $a_n(p)$ is a penalty term.

2.3 Convergence of the penalized likelihood estimate

The convergence result is based on the following inequality, which is the immediate multivariate case generalization of the inequality in Gassiat (2002) :

Proposition 1 *: Let $\mathcal{G} \subset \mathcal{G}_P$ be a parametric family of conditional densities containing the true model f and let us define the generalized score function*

$$s_g(y_1, y_2) = \frac{\frac{g(y_1,y_2)}{f(y_1,y_2)} - 1}{\left\| \frac{g}{f} - 1 \right\|_{L^2(\mu)}}$$

where μ is the stationary measure of (Y_{k-1}, Y_k). Then,

$$sup_{g \in \mathcal{G}} \left(l_n(g) - l_n(f) \right) \leq \frac{1}{2} sup_{g \in \mathcal{G}} \frac{\left(\sum_{k=2}^{n} s_g(y_{k-1}, y_k) \right)^2}{\sum_{k=2}^{n} (s_g)_-^2 (y_{k-1}, y_k)}$$

with $(s_g)_-(y_{k-1}, y_k) = min(0,\, s_g(y_{k-1}, y_k))$.

Now we can state the following convergence theorem which generalizes the result in Gassiat (2002) :

Theorem 1 *: Consider the model $(Y_k,\, X_k)$ defined by (1) and the penalized-likelihood criterion introduced in (2). Let us introduce the next assumptions :*

(A1) *$a_n(\cdot)$ is an increasing function of p, $a_n(p_1) - a_n(p_2) \to \infty$ when $n \to \infty$ for every $p_1 > p_2$ and $\frac{a_n(p)}{n} \to 0$ when $n \to \infty$ for every p*

(A2) *the model $(Y_k,\, X_k)$ verifies the weak identifiability assumption* ***(HI)***

$$\sum_{i=1}^{p} \pi_i f_i(y_2 - F_i(y_1)) = \sum_{i=1}^{p_0} \pi_i^0 f_i^0 \left(y_2 - F_i^0(y_1) \right) \Leftrightarrow \sum_{i=1}^{p} \pi_i \delta_{\theta_i} = \sum_{i=1}^{p_0} \pi_i^0 \delta_{\theta_i^0},$$

where δ_{θ_i} is the Dirac measure.

(A3) *the parameterization $\theta_i \to f_i(y_2 - F_i(y_1))$ is continuous for every (y_1, y_2) and there exists $m(y_1, y_2)$ an integrable map with respect to the stationary measure of (Y_k, Y_{k-1}) such that $|log(g)| < m$*

(A4) *Y_k satisfies the hypothesis* ***(HS)*** *and the family of generalized score functions associated to $\mathcal{G}_P$*

$$\mathcal{S} = \left\{ s_g,\ s_g\left(y_1, y_2\right) = \frac{\frac{g(y_1,y_2)}{f(y_1,y_2)} - 1}{\left\| \frac{g}{f} - 1 \right\|_{L^2(\mu)}},\ g \in \mathcal{G}_P,\ g \neq f \right\} \subset \mathcal{L}_2\left(\mu\right)$$

and for every $\varepsilon > 0$

$$\mathcal{H}_{[]}\left(\varepsilon, \mathcal{S}, \|\cdot\|_2\right) = \mathcal{O}\left(\left|log\,\varepsilon\right|\right),$$

where $\mathcal{H}_{[]}\left(\varepsilon, \mathcal{S}, \|\cdot\|_2\right)$ *is the bracketing entropy of* $\mathcal{S}$ *with respect to the* L_2*-norm. Then, under the hypothesis (A1)-(A4) and (HC),* $\hat{p} \to p_0$ *in probability.*

For parcimony purposes, the proof is omitted here. The complete proof, as well as some definitions of emipirical processes for dependent data, bracketing entropy and Donsker classes are available in Olteanu and Rynkiewicz (2006).

3 Application for Gaussian noise

This section is devoted to verifying the hypothesis **(A1)-(A4)** of Theorem 1 in the case of a normally distributed noise. Let us then consider the model defined in (1) such that, for every $i \in \{1, ..., p_0\}$, $f_i^0 \sim \mathcal{N}\left(0, \left(\sigma_i^0\right)^2\right)$. The parameter corresponding to one component of the true model, i, will be denoted by $\theta_i^0 = \left(a_i^0, b_i^0, \sigma_i^0\right)$.

Sufficient conditions for strict stationarity and ergodicity are given by the next proposition.

Proposition 2 *: If* $\left|a_i^0\right| < 1$ *for every* $i \in \{1, ..., p_0\}$*, then* (X_t, Y_t) *is strictly stationary, geometrically ergodic and, in particular, geometrically* β*-mixing. Moreover, there exists* $\delta > 0$ *such that* $E\left(e^{\delta Y_t^2}\right) < \infty$.

The estimate of the number of components p_0 is then constructed as follows : let us consider a maximum number of regimes $P > 0$ and the class of all possible conditional densities of Y_t marginal in X_t :

$$\mathcal{G}_P = \bigcup_{p=1}^{P} \mathcal{G}_p\,,\ \mathcal{G}_p = \left\{ g \mid g\left(y_1, y_2\right) = \sum_{i=1}^{p} \pi_i f_i\left(y_2 - F_i\left(y_1\right)\right) \right\}$$

where

- $\sum_{i=1}^{p} \pi_i = 1$ and, with no loss of generality, we suppose that for every $i \in \{1, ..., p\}$, $\pi_i \geq \eta > 0$
- for every $i \in \{1, ..., p\}$, $F_i\left(y\right) = a_i y + b_i$, $f_i \sim \mathcal{N}\left(0, \sigma_i^2\right)$ and $\theta_i = (a_i,\, b_i,\, \sigma_i)$ belongs to a compact set

Then, the estimate $\hat{p}$ is defined as the maximizer of (2) and it converges in probability to the true number of regimes if the assumptions of Theorem 1 hold. The key hypothesis is that the class of generalized score functions

$$\mathcal{S} = \left\{ s_g,\ s_g = \frac{\frac{g}{f} - 1}{\left\| \frac{g}{f} - 1 \right\|_{L^2(\mu)}},\ g \in \mathcal{G}_P,\ \left\| \frac{g}{f} - 1 \right\|_{L^2(\mu)} \neq 0 \right\}$$

is Donsker. First, we shall verify that this class is well defined, that is $\left\| \frac{g}{f} - 1 \right\|_{L^2(\mu)} < \infty$, for all $g \in \mathcal{G}_P$. The next statement gives sufficient conditions for the existence of the generalized score functions .

Proposition 3 *:* $\left\| \frac{g}{f} - 1 \right\|_{L^2(\mu)} < \infty$ *if for every* $i \in \{1, ..., p\}$*, there exists* $k \in \{1, ..., p_0\}$ *such that* $\sigma_i^2 < 2\left(\sigma_k^0\right)^2$ *and* $\left|a_i - a_k^0\right| < \sqrt{\delta\left(2\left(\sigma_k^0\right)^2 - \sigma_i^2\right)}$ *for* $\delta > 0$ *verifying* $E\left(e^{\delta Y_t^2}\right) < \infty$.

According to Teicher (1963), the weak identifiability hypothesis **(A2)** is verified for mixtures of Gaussian densities. Moreover, since, by assumption, for every $i \in \{1, ..., p\}$, $\pi_i \geq \eta > 0$, the estimates $\hat{\theta}_n = \left(\hat{\theta}_{1,n}, ..., \hat{\theta}_{p,n}\right)$ are consistent and the sufficient conditions in Proposition 3 are verified immediately for n sufficiently large.

Next, let us prove that $\mathcal{S}$ is Donsker and that $\mathcal{H}_{[]}\left(\varepsilon, \mathcal{S}, \|\cdot\|_2\right) = \mathcal{O}\left(|log\,\varepsilon|\right)$ for all $\varepsilon > 0$. For $g \in \mathcal{G}_P$, let us denote $\theta = (\theta_1, ..., \theta_p)$ and $\pi = (\pi_1, ..., \pi_p)$, so that the global parameter will be $\Phi = (\theta, \pi)$ and the associated generalized score function

$$s_\Phi := s_g = \frac{\frac{g}{f} - 1}{\left\| \frac{g}{f} - 1 \right\|_{L^2(\mu)}}$$

Proving that a parametric family like $\mathcal{S}$ is a Donsker class is usually immediate under good regularity conditions (see, for instance, Van der Vaart, 2000). In this particular case, the problems arise when $g \to f$ and the limits in $L^2(\mu)$ of s_g have to be computed. To achieve checking the hypothesis **(A4)**, let us then split $\mathcal{S}$ into two classes of functions. We shall consider $\mathcal{F}_0 \subset \mathcal{G}_P$ a neighbourhood of f such that it exists $\delta > 0$ verifying $\mathcal{F}_0 = \left\{ g \in G_p,\ \left\| \frac{g}{f} - 1 \right\|_{L^2(\mu)} \leq \delta,\ g \neq f \right\}$ and let $\mathcal{S}_0 = \{s_g,\ g \in \mathcal{F}_0\}$.

On $\mathcal{S} \setminus \mathcal{S}_0$, it can be easily seen that

$$\left\| \frac{\frac{g_1}{f} - 1}{\left\| \frac{g_1}{f} - 1 \right\|_{L^2(\mu)}} - \frac{\frac{g_2}{f} - 1}{\left\| \frac{g_2}{f} - 1 \right\|_{L^2(\mu)}} \right\|_{L^2(\mu)} \leq \frac{2}{\delta} \left\| \frac{g_1}{f} - \frac{g_2}{f} \right\|_{L^2(\mu)}$$

On the other hand, under the assumptions in Proposition 3, $\frac{g}{f}$ has square integrable partial derivatives of order one and, using the result on parametric classes of functions in Van der Vaart (2000), we get that

$$\mathcal{N}_{[]}\left(\varepsilon, \mathcal{S} \setminus \mathcal{S}_0, \|\cdot\|_2\right) = \mathcal{O}\left(\frac{1}{\delta\varepsilon}\right)^{4P},$$

where $\mathcal{N}_{[]}\left(\varepsilon, \mathcal{S} \setminus \mathcal{S}_0, ||\cdot||_2\right)$ is the number of ε-brackets necessary to cover $\mathcal{S} \setminus \mathcal{S}_0$ and the bracketing entropy is computed as $\mathcal{H}_{[]}\left(\varepsilon, \mathcal{S} \setminus \mathcal{S}_0, ||\cdot||_2\right) = log\mathcal{N}_{[]}\left(\varepsilon, \mathcal{S} \setminus \mathcal{S}_0, ||\cdot||_2\right)$.

It remains to prove that $\mathcal{S}_0$ is Donsker. The guiding idea is to reparameterize the model in a convenient manner which will allow a Taylor expansion around the identifiable part of the true value. For that, we shall use a slight modification of the method proposed by Liu and Shao (2003).

In the following we will make the additional assumption $p_0 < p$. Let us remark that when $\frac{g}{f} - 1 = 0$, the weak identifiability hypothesis **(A2)** and the fact that for every $i \in \{1, ..., p\}$, $\pi_i \geq \eta > 0$, implies that there exists a vector $t = (t_i)_{0 \leq i \leq p_0}$ such that $0 = t_0 < t_1 < ... < t_{p_0} = p$ and, modulo a permutation, Φ can be rewritten as follows :

$$\theta_{t_{i-1}+1} = ... = \theta_{t_i} = \theta_i^0,\ \sum_{j=t_{i-1}+1}^{t_i} \pi_j = \pi_i^0,\ i \in \{1, ..., p_0\}$$

With this remark, one can define in the general case $s = (s_i)_{1 \leq i \leq p_0}$ and $q = (q_j)_{1 \leq j \leq p}$ so that, for every $i \in \{1, ..., p_0\}$, $j \in \{t_{i-1} + 1, ..., t_i\}$,

$$s_i = \sum_{j=t_{i-1}+1}^{t_i} \pi_j - \pi_i^0,\ q_j = \frac{\pi_j}{\sum_{l=t_{i-1}+1}^{t_i} \pi_l}$$

and the new parameterization will be

$$\Theta_t = (\phi_t, \psi_t)\,,\ \phi_t = \left((\theta_j)_{1 \leq j \leq p}\,, (s_i)_{1 \leq i \leq p_0 - 1}\right),\ \psi_t = (q_j)_{1 \leq j \leq p}$$

with ϕ_t containing all the identifiable parameters of the model and ψ_t the non-identifiable ones. Then, for $g = f$, we will have that

$$\phi_t^0 = (\underbrace{\theta_1^0, ..., \theta_1^0}_{t_1}\ ,...,\ \underbrace{\theta_{p_0}^0, ..., \theta_{p_0}^0}_{t_{p_0} - t_{p_0 - 1}},\ \underbrace{0, ..., 0}_{p_0 - 1})^T$$

This reparameterization allows to write a second-order Taylor expansion of $\frac{g}{f} - 1$ at ϕ_t^0. For ease of writing, we shall first denote

$$g_j\left(y_1, y_2\right) = g_{\theta_j}\left(y_1, y_2\right) = \frac{f_j(y_2 - F_j(y_1))}{\sum_{i=1}^{p_0} \pi_i^0 f_i^0\left(y_2 - F_i^0(y_1)\right)} - 1$$

Then, the density ratio becomes :

$$\frac{g}{f} - 1 = \sum_{i=1}^{p_0 - 1} \left(s_i + \pi_i^0\right) \sum_{j=t_{i-1}+1}^{t_i} q_j g_j + \left(\pi_{p_0}^0 - \sum_{i=1}^{p_0 - 1} s_i\right) \sum_{j=t_{p_0-1}+1}^{t_{p_0}} q_j g_j$$

By remarking that when $\phi_t = \phi_t^0$, $\frac{g}{f}$ does not vary with ψ_t, we will study the variation of this ratio in a neighbourhood of ϕ_t^0 and for fixed ψ_t. First, let us introduce the following notations of the ϕ_t-derivatives of g_j computed at ϕ_t^0:

$$g_j' := \frac{\partial g_j}{\partial \theta_j}\left(\phi_t^0, \psi_t\right),\ g_j'' := \frac{\partial^2 g_j}{\partial \theta_j^2}\left(\phi_t^0, \psi_t\right),\ g_j''' := \frac{\partial^3 g_j}{\partial \theta_j^3}\left(\phi_t^0, \psi_t\right)$$

With these notations we can state the following result :

Proposition 4 *: Let us denote* $D(\phi_t, \psi_t) = \left\| \frac{g_{(\phi_t,\psi_t)}}{f} - 1 \right\|_{L^2(\mu)}$. *For any fixed* ψ_t, *there exists the second-order Taylor expansion at* ϕ_t^0 *:*

$$\frac{g}{f} - 1 = \left(\phi_t - \phi_t^0\right)^T g'_{(\phi_t^0,\psi_t)} + \frac{1}{2}\left(\phi_t - \phi_t^0\right)^T g''_{(\phi_t^0,\psi_t)}\left(\phi_t - \phi_t^0\right) + o\left(D\left(\phi_t, \psi_t\right)\right)$$

$$\left(\phi_t - \phi_t^0\right)^T g'_{(\phi_t^0,\psi_t)} = \sum_{i=1}^{p_0} \pi_i^0 \left(\sum_{j=t_{i-1}+1}^{t_i} q_j \theta_j - \theta_i^0 \right)^T g_i' + \sum_{i=1}^{p_0} s_i g_{\theta_i^0}$$

$$\left(\phi_t - \phi_t^0\right)^T g''_{(\phi_t^0,\psi_t)}\left(\phi_t - \phi_t^0\right) = \sum_{i=1}^{p_0} \left[2 s_i \left(\sum_{j=t_{i-1}+1}^{t_i} q_j \theta_j - \theta_i^0 \right)^T g_i' + \right.$$

$$\left. + \pi_i^0 \sum_{j=t_{i-1}+1}^{t_i} q_j \left(\theta_j - \theta_i^0\right)^T g_i'' \left(\theta_j - \theta_i^0\right) \right]$$

Moreover,

$$\left(\phi_t - \phi_t^0\right)^T g'_{(\phi_t^0,\psi_t)} + \frac{1}{2}\left(\phi_t - \phi_t^0\right)^T g''_{(\phi_t^0,\psi_t)}\left(\phi_t - \phi_t^0\right) = 0 \Leftrightarrow \phi_t = \phi_t^0$$

Using the Taylor expansion above, we can now show that $\mathcal{S}_{\mathcal{F}_0} = \{s_g,\, g \in \mathcal{F}_0,\, g \neq f\}$ is a Donsker class. The result is stated in the following :

Proposition 5 *: The number of* ε*-brackets* $\mathcal{N}_{[]}\left(\varepsilon, \mathcal{S}_0, \|\cdot\|_2\right)$ *covering* $\mathcal{S}_0$ *is* $\mathcal{O}\left(\frac{1}{\varepsilon}\right)^{9p_0}$.

With this last assertion, it is proven that Theorem 1 applies in the Gaussian case and that the only constraints are the stationarity of each autoregressive model in the mixture and the choice of a penalty term according to hypothesis **(A1)**.

4 Numerical examples

Once the theoretical result is verified in the Gaussian case, let us give some numerical examples to illustrate it. Three things will be interesting to study: the speed of convergence, since we do not have it theoretically, the stability and the influence of the penalty term. For parcimony purposes and because of the important computation time, only the BIC penalty term was considered here

$$a_n(p) = \frac{1}{2} k(p) \, ln(n),$$

where $k(p)$ is the number of parameters of a model with p components and n is the size of the sample.

The examples are mixtures of two autoregressive models in which we vary the leading coefficients and the weights of the discrete mixing distribution. For each of them, we simulate 20 samples of lenghts $n = 200, 500, 1000, 1500, 2000$ and we fix $P = 3$ the upper bound for the number of regimes.

We look for $p \in \{1, 2, 3\}$ which maximizes the criterion $T_n(p) = sup_{g \in \mathcal{G}_p} l_n(g) - a_n(p)$. We suppose in the following that the number of time lags is known and that the regression functions are linear. At p fixed, the likelihood is maximized via an EM algorithm (see, for instance, Dempster, Laird and Rubin, 1977 or Redner and Walker, 1984). As our goal is to estimate the number of components, we shall not insist on the other parameters estimated by the EM algorithm. Its detailed version for a fixed number of components is available in Wong and Li (2000).

To avoid local maxima, the procedure is initialized several times with different starting values : in our case, ten different initializations provided good results. The stopping criteria applies when either there is no improvement in the likelihood value, either a maximum number of iterations, fixed at 200 here for reasonable computation time, is reached. The results are summarized in Tables 1 and 2 at the end of this paper. The true conditional density is

$$f(y_1, y_2) = \pi_1^0 f_1^0 \left(y_2 - F_1^0(y_1)\right) + \left(1 - \pi_1^0\right) f_2^0 \left(y_2 - F_2^0(y_1)\right)$$

with $F_i^0(y_1) = a_i^0 y_1 + b_i^0$ and $f_i^0 \sim \mathcal{N}\left(0, \left(\sigma_i^0\right)^2\right)$ for $i \in \{1, 2\}$. For every example, we pick equal standard errors $\sigma_1^0 = \sigma_2^0 = 0.5$ and let vary the rest of the coefficients: $\pi_1^0 \in \{0.5, 0.7, 0.9\}$, $a_1^0, a_2^0 \in \{0.1, 0.5, 0.9\}$, $b_1^0 \in \{1, 0.5\}$ and $b_2^0 \in \{-1, -0.5\}$. In Table 1, the convergence is reached rapidly for a small number of observations, while in Table 2 this is less obvious, since the two components are chosen closer. However, in most of the examples, 2000 sample points are enough to obtain a good estimate of the number of regimes.

5 Conclusion and future work

We proved the convergence of penalized likelihood criteria for estimating the number of components in a mixture autoregressive model. The hypothesis of the main result were shown to be verified in the Gaussian case and some numerical examples illustrated the convergence properties. Using the BIC criterion for selecting the number of components is thus theoretically justified. However, several questions arise at this state and represent future research directions. In our opinion, there are two important generalizations to be studied : on one hand, consider the case where the number of time lags is also unknown and build an estimate for both the number of components and the number of lags and, on the other hand, consider the more general frame of autoregressive Markov switching models and estimate the number of hidden states.

References

[1] Bollerslev T. (1986) Generalized autoregressive conditional heteroscedasticity, *Journal of Econ.*, 31, 307-327

[2] Dacunha-Castel le D., Gassiat E. (1997) The estimation of the order of a mixture model, *Bernoulli*, 3, 279-299

[3] Dacunha-Castelle D., Gassiat E. (1997) Testing in locally conic models, ESAIM Prob. and Stat., 1, 285-317

[4] Dempster A.P., Laird N.M., Rubin D.B (1977) Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statist. Soc. (B)*, 39(1), 1-38

[5] Engle R.F. (1982) Autoregressive conditional heteroscedasticity with estimates of the variance of U.K. inflation, *Econometrica*, 50, 987-1008

[6] Gassiat E. (2002) Likelihood ratio inequalities with applications to various mixtures, *Ann. Inst. Henri Poincaré*, 38, 897-906

[7] Hamilton J.D. (1989) A new approach to the economic analysis of nonstationary time series and the business cycle, *Econometrica*, 57, 357-384

[8] Henna J. (1985) On estimating the number of constituents of a finite mixture of continuous distributions, *Ann. Inst. Statist. Math.*, 37, 235-240

[9] Izenman A.J., Sommer C. (1988) Philatelic mixtures and multivariate densities, *Journal of the American Stat. Assoc.*, 83, 941-953

[10] Keribin C. (2000) Consistent estimation of the order of mixture models, *Sankhya : The Indian Journal of Statistics*, 62, 49-66

[11] Leroux B.G. (1992) Maximum penalized likelihood estimation for independent and Markov-dependent mixture models, *Biometrics*, 48, 545-558

[12] Lindsay B.G. (1983) Moment matrices : application in mixtures, *The Annals of Statistics*, 17, 722-740

[13] Liu X., Shao Y. (2003) Asymptotics for likelihood ratio tests under loss of identifiability, *The Annals of Statistics*, 31(3), 807-832

[14] Olteanu M., Rynkiewicz J. (2006) Estimating the number of regimes in a switching autoregressive model, *Prepub. SAMOS 245*, http://samos.univ-paris1.fr

[15] Redner R.A., Walker H.F. (1984) Mixture densities, maximum likelihood and the EM algorithm, *SIAM Review*, 26(2), 195-239

[16] Roeder K. (1994) A graphical technique for determining the number of components in a mixture of normals, *Journal of the American Stat. Assoc.*, 89, 487-495

[17] Teicher H. (1963) Identifiability of finite mixtures, *Ann. Math. Statist.*, 34(2), 1265-1269

[18] Tong H. (1983) *Threshold models in non-linear time series analysis*, Lecture Notes in Statistics, 21, Springer, Heidelberg

[19] Van der Vaart A.W. (2000) *Asymptotic Statistics*, Cambridge University Press

[20] Wong C.H., Li W.K. (2000) On a mixture autoregressive model, *J. of Royal Stat. Soc (Series B)*, 62, 95-115

[21] Yao J.F., Attali J.G. (2000) On stability of nonlinear AR processes with Markov switching, *Advances in Applied Probability*, 32 (2), 394-407

	π_1^0	0.5			0.7			0.9		
	n	$\hat{p}=1$	$\hat{p}=2$	$\hat{p}=3$	$\hat{p}=1$	$\hat{p}=2$	$\hat{p}=3$	$\hat{p}=1$	$\hat{p}=2$	$\hat{p}=3$
$a_1^0=0.1$	200	0	20	0	0	20	0	0	18	2
$a_2^0=0.1$	500	0	20	0	0	20	0	0	20	0
	1000	0	20	0	0	20	0	0	20	0
	1500	0	20	0	0	20	0	0	20	0
	2000	0	20	0	0	20	0	0	20	0
$a_1^0=0.1$	200	0	20	0	0	19	1	1	19	0
$a_2^0=0.5$	500	0	20	0	0	20	0	0	20	0
	1000	0	20	0	0	20	0	0	20	0
	1500	0	20	0	0	20	0	0	20	0
	2000	0	20	0	0	20	0	0	20	0
$a_1^0=0.1$	200	0	20	0	0	20	0	4	16	0
$a_2^0=0.9$	500	0	20	0	0	20	0	0	20	0
	1000	0	20	0	0	20	0	0	20	0
	1500	0	20	0	0	20	0	0	20	0
	2000	0	20	0	0	20	0	0	20	0
$a_1^0=0.5$	200	0	19	1	0	18	2	0	20	0
$a_2^0=0.5$	500	0	20	0	0	20	0	0	18	2
	1000	0	20	0	0	20	0	0	20	0
	1500	0	20	0	0	20	0	0	20	0
	2000	0	20	0	0	20	0	0	20	0
$a_1^0=0.5$	200	0	19	1	0	20	0	11	9	0
$a_2^0=0.9$	500	0	20	0	0	20	0	0	20	0
	1000	0	20	0	0	20	0	0	20	0
	1500	0	20	0	0	20	0	0	20	0
	2000	0	20	0	0	20	0	0	20	0
$a_1^0=0.9$	200	0	20	0	0	20	0	0	16	4
$a_2^0=0.9$	500	0	20	0	0	20	0	0	20	0
	1000	0	19	1	0	20	0	0	20	0
	1500	0	20	0	0	20	0	0	20	0
	2000	0	20	0	0	20	0	0	20	0

Table 1: Results for $b_1^0=1$, $b_2^0=-1$, $\sigma_1^0=\sigma_2^0=0.5$

	π_1^0	0.5			0.7			0.9		
	n	$\hat{p}=1$	$\hat{p}=2$	$\hat{p}=3$	$\hat{p}=1$	$\hat{p}=2$	$\hat{p}=3$	$\hat{p}=1$	$\hat{p}=2$	$\hat{p}=3$
$a_1^0=0.1$	200	20	0	0	20	0	0	20	0	0
$a_2^0=0.1$	500	18	2	0	18	2	0	20	0	0
	1000	14	6	0	9	11	0	11	9	0
	1500	6	14	0	4	16	0	5	15	0
	2000	5	15	0	0	20	0	1	19	0
$a_1^0=0.1$	200	12	8	0	13	7	0	20	0	0
$a_2^0=0.5$	500	11	19	0	6	14	0	18	2	0
	1000	0	20	0	1	19	0	14	6	0
	1500	0	20	0	0	20	0	8	12	0
	2000	0	20	0	0	20	0	7	13	0
$a_1^0=0.1$	200	0	20	0	4	16	0	17	3	0
$a_2^0=0.9$	500	0	20	0	0	20	0	9	11	0
	1000	0	20	0	0	20	0	9	11	0
	1500	0	20	0	0	20	0	4	16	0
	2000	0	20	0	0	20	0	0	20	0
$a_1^0=0.5$	200	18	2	0	20	0	0	19	1	0
$a_2^0=0.5$	500	20	0	0	19	1	0	19	1	0
	1000	14	6	0	13	7	0	10	10	0
	1500	9	11	0	5	15	0	5	15	0
	2000	3	17	0	0	20	0	3	17	0
$a_1^0=0.5$	200	9	11	0	11	9	0	20	0	0
$a_2^0=0.9$	500	0	20	0	7	13	0	19	1	0
	1000	0	20	0	0	20	0	19	1	0
	1500	0	20	0	0	20	0	18	2	0
	2000	0	20	0	0	20	0	14	6	0
$a_1^0=0.9$	200	20	0	0	19	1	0	19	1	0
$a_2^0=0.9$	500	20	0	0	18	2	0	17	3	0
	1000	14	6	0	7	13	0	11	9	0
	1500	7	13	0	5	15	0	3	17	0
	2000	6	14	0	0	20	0	0	20	0

Table 2: Results for $b_1^0=0.5$, $b_2^0=-0.5$, $\sigma_1^0=\sigma_2^0=0.5$

Incorporating Seasonal Information on Direct and Recursive Predictors Using LS-SVM

L. J. Herrera, H. Pomares, I. Rojas, A.Guillén, G. Rubio

University of Granada - Dept of Computer Architecture and Computer Technology
Granada- Spain

Abstract. In many time series prediction problems, the only data available is a single series of data. However, in the case of time series that present a certain seasonality, the long term prediction capabilities of a predictor can be improved by considering that seasonality as an additional series of input data. This paper deals with the new benchmark proposed in the ESTSP'2007 conference, and presents a solution based on Least Squares Support Vector Machines (LS-SVMs), which specially bring very good performance on time series prediction and function approximation problems. The seasonality of the time series will be covered in the LS-SVM model by adding an additional input variable that identifies the time step inside a season. Both direct and recursive predictors will be evaluated with this technique, using also a simple wrapper approach to perform variable selection.

1 Introduction

Time series forecasting is a challenge in many fields [1]. It is a complex problem, that in general has several points in common with function approximation problems. The analysis of time series is based on the assumption that successive values in the data file represent consecutive measurements taken at equally spaced time intervals. There are two main goals of time series analysis: (a) identifying the nature of the phenomenon represented by the sequence of observations, and (b) forecasting (predicting future values of the time series variable). Those goals are normally joined by approaching the problem as a modeling I/O data task, which on the one hand, builds up a model that tries to identify the internal structure of the observed data, in order to, on the other hand, predict novel values of the series using that predictive model.

Long term predictions require a more careful study, since long term data dependencies are more difficult to model. Two approaches can be taken to tackle this problem: direct prediction and recursive prediction [2]. Variable selection is a very important sub-task in any modeling problem. Specially in time series prediction problems, in which the input space is a priori unknown, and any number of previous time steps ($x(t-0)$, $x(t-1)$, ..., $x(t-\tau)$) are candidate input variables to predict the value of the series at the time step $t+h$ [9] in the general model

$$\hat{x}(t+h) = F\left(x\left(t-0\right),\ x\left(t-1\right), \ldots\ x\left(t-\tau\right)\right) \tag{1}$$

A good analysis of the time series is necessary to properly tackle the modeling and prediction problems. For example, when it is detected that the series presents a certain seasonality, this seasonality has to be included in the prediction model [10, 11].

This work proposes a solution to the new time series prediction problem provided in the ESTSP'2007 conference. The solution uses Least Squares-Support Vector Machines (LS-SVM) [6], and includes the seasonal information present in the time series by adding the time step within the season, of the value to be predicted, as an additional input variable to the general prediction model in equation 1. This modified general prediction model using LS-SVM will be evaluated using both recursive prediction and direct prediction. Furthermore, variable selection will be performed using a simple wrapper approach.

The rest of the work is structured as follows. Section 2 briefly reviews the LS-SVM learning methodology. Section 3 reviews the concepts of recursive prediction and direct prediction for long term time series forecasting. Section 4 discusses problem of variable selection, discussing the variable selection strategy performed in the work. Section 5 analyzes the ESTST'2007 benchmark, and addresses the seasonality present in the series of data samples. Section 6 presents the results obtained in he prediction, and section 7 concludes the paper.

2 Least Squares Support Vector Machines

LS-SVMs are reformulations to standard SVMs that lead to solving linear Karush-Kuhn-Tucker (KKT) systems [5]. LS-SVMs are regularized supervised approximators, closely related to regularization networks and Gaussian processes, but that additionally emphasize and exploit primal-dual interpretations from optimization theory [6].

The LS-SVM model [6] is defined in its primal weight space by

$$\hat{y} = \vec{\omega}^T \phi(\vec{x}) + b \tag{2}$$

where $\vec{\omega}^T$ and b are the parameters of the model, $\phi(\vec{x})$ is a function that maps the input space into a higher dimensional feature space, and $\vec{x}$ is the n-dimensional vector of inputs x_j. In Least Squares Support Vector Machines for function approximation, the following optimization problem is formulated,

$$\min_{\vec{\omega},b,e} J(\omega, e) = \frac{1}{2}\vec{\omega}^T\vec{\omega} + \gamma\frac{1}{2}\sum_{i=1}^{N} e_i^2 \tag{3}$$

subject to the equality constraints (inequality constraints in the case of SVMs)

$$e_i = y_i - \hat{y}(\vec{x}_i), i = 1 \ldots N \tag{4}$$

Solving this optimization problem in dual space, leads to finding the λ_i and b coefficients in the following solution

$$\hat{y} = \sum_{i=1}^{N} \lambda_i K(\vec{x}, \vec{x}_i) + b \tag{5}$$

where the function $K(\vec{x}, \vec{x}_i)$ is the kernel function defined as the dot product between the $\phi(\vec{x})$ and $\phi(\vec{x}_i)$ mappings.

In case we consider Gaussian kernels, the kernel function $K(\vec{x}, \vec{x}_i)$ takes the form

$$K(\vec{x}, \vec{x}_i) = \exp\left[-\left(\frac{||\vec{x} - \vec{x}_i||}{\sqrt{2}\sigma_i}\right)^2\right] \tag{6}$$

where σ_i is the width of the kernel, that together with the regularization parameter γ, are the hyper-parameters of the problem. Note that in the case in which Gaussian kernels are used, the models obtained resemble Radial Basis Function Networks (RBFN); with the particularities that there is an RBF node per data point, and that overfitting is controlled by a regularization parameter instead of by reducing the number of kernels [6]. In LS-SVM, the hyper-parameters of the model can be optimized by cross-validation. Nevertheless, in order to speed-up the optimization, a more efficient methodology by Lendasse et al. can be found in [3]. A Matlab toolbox for LS-SVMs can be found in [4].

LS-SVMs present a very good performance for function approximation and time series prediction problems [12]. From the computational complexity point of view, LS-SVMs don't suffer from the curse of dimensionality in the number of input dimensions, but in the number of training data points. This makes LS-SVMs to be easier to apply to complex problems in which the input space is high-dimensional.

3 Recursive and Direct Prediction

In general, long term time series prediction is a more complex task since the uncertainty increases with the horizon of prediction. In this case, two trends can be taken to tackle the modeling and prediction problem: direct prediction and recursive prediction [2]. Direct prediction implies the construction of different models, one for each different prediction horizon needed. On the other side, recursive prediction only uses one model to predict all the horizons needed. Due to the characteristics of each approach, it can be expected that direct prediction provides a better performance in prediction accuracy, since it uses one specific model for each desired horizon, and in recursive prediction the error committed in nearest horizons can be transmitted to further horizons. Nevertheless direct prediction has the drawback that too many models might be needed to obtain the long term prediction [7, 8], and the time series behaviour understandability vanishes as several different models are needed.

This work considers both opposite approaches in order to verify their limitations, advantages and performance.

4 Wrapper Variable Selection

The goal of feature selection and dimension reduction in general is twofold. First, reducing the number of input variables fights the curse of dimensionality, and gives the possibility of increasing the generalization performance of the model. Second, the interpretation of the relationship between features and outputs in the designed model also improves [6].

Several approaches exist in the literature to perform variable selection. Filter methods try to select the variables in a preprocess step with the only information that the I/O values provide. Wrapper methods employ the learning methodology that is going to be used, in order to select the subset of variables that brings the best performance. Filter methods have the advantage that the learning methodology used is not considered, and the selection procedure is faster. However, wrapper methods have the advantage that they guarantee the performance of the learning methodology with the subset of variables considered.

In this work, the variable selection method performed is a wrapper method, that selects the number τ of previous time steps $x(t-0)$, $x(t-1)$, ..., $x(t-\tau)$, considered as input variables in the general model 1. The presence of the additional variable $s(t+h)$, indicating the time step within the season of the horizon to be predicted h, is also evaluated together with the selection of τ. The different values of τ and the consideration of $s(t+h)$ is evaluated by optimizing different LS-SVMs, using part of the training data set (4/5) and testing it with the remaining data samples (1/5). In order to decrease the computational demand of the wrapper procedure, the LS-SVM were optimized using a simple 4x4 grid search 4-fold cross-validation optimization. This simple but effective procedure assures that a suboptimal subset of variables is used in the learned model.

5 ESTSP'2007 Benchmark. Data Analysis

The data set provided is shown in *fig.* 1. The number of samples in the time series is 875. The goal of the competition is the prediction of the 50 next values of the time series. The evaluation of the performance will be done using the MSE obtained from the prediction of both the 15 and the 50 next values.

In order to properly work with the time series using LS-SVM, te data series was normalized to have zero mean and unit variance. From the 875 data samples of the series, the first 800 data samples were used as training data, leaving the remaining 75 data samples as test data.

5.1 Seasonality in time series

Seasonality in time series occurs when there is a certain pattern that is repeated each k elements. The data series in *fig.* 1 shows to have a certain seasonality, as can be seen from the autocorrelation function of the time series in *fig.* 2.

The period of the seasonality can be obtained by a number of techniques. In this work, the period has been obtained by evaluating the distances between the supposed-seasons data. The specific period for which the sum of squared

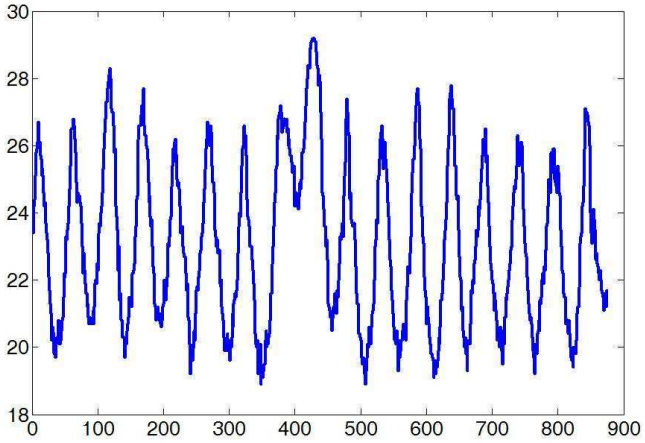

Fig. 1: Original data set

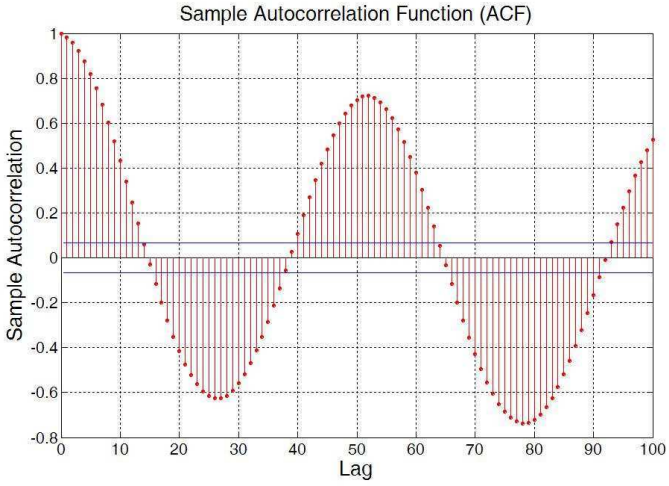

Fig. 2: Sample Autocorrelation Function

distances among the different seasons data was lowest, was taken as a solution. The specific period found in this case was 52; *fig.* 3 shows the superposition of the different seasons of 52 data samples of the time series. This value reminds a weekly registered phenomena along different years (seasons). In the LS-SVM model, as it was explained in the previous section, a variable $s(t+h)$ (indicating the time step within the season of the horizon to be predicted h) is added as a new input variable to improve the prediction accuracy. Therefore this variable will take values from 1..52 in this application.

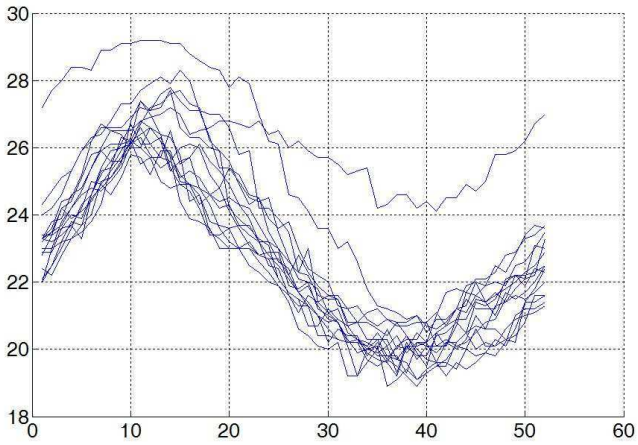

Fig. 3: Different seasons on the original time series

6 Simulations

6.1 Recursive prediction

In recursive long term prediction, as was discussed, the outputs of the model are subsequently used as inputs in the form

$$\hat{x}(t+1) = F\left(s(t+h), \hat{x}\left(t-0\right),\ \hat{x}\left(t-1\right), \ldots\ \hat{x}\left(t-\tau\right)\right) \tag{7}$$

where the first input values $\hat{x}\left(t-0\right)$, $\hat{x}\left(t-1\right)$, $\ldots$, $\hat{x}\left(t-\tau\right)$ are known.

The construction of a recursive predictor in this case takes $h = 1$, and is trained similarly as is done for a direct predictor with the same horizon. However the performance of the recursive model has to be evaluated, by recursively applying the model to predict all the horizons needed. In this problem the evaluation is done for 50 consecutive values using the same model (see section 5).

6.1.1 Variable Selection

The variable selection procedure is performed for different values of τ, evaluating too the inclusion of $s(t+h)$ as an additional variable. The training of the LS-SVMs is performed using grid-search cross-validation (see section 4). The training and validation data sets are distributed as mentioned in section 4.

Table 1 shows the training and test MSE when performing the variable selection procedure for different values of τ, when taking and not taking into account the step in the season $s(t+1)$. The MSE error values were obtained by averaging the evaluation all possible recursive predictions of 50 values both in the training and validation data sets.

τ	with $s(t+1)$ training MSE	with $s(t+1)$ validation MSE	without $s(t+1)$ training MSE	without $s(t+1)$ validation MSE
$\tau = 0$	1.2940	0.5702	10.1460	9.2555
$\tau = 1$	1.2690	0.4761	9.1123	8.4664
$\tau = 2$	1.2894	0.5263	7.9205	7.4484
$\tau = 3$	1.2596	0.5100	6.4598	5.6268
$\tau = 4$	1.2520	0.5028	5.1546	3.2159
$\tau = 5$	1.2371	0.5454	3.7612	1.8921
$\tau = 6$	1.2941	0.4807	3.5651	1.5770
$\tau = 7$	1.2332	0.4965	2.8299	1.1174
$\tau = 8$	1.2944	0.4672	3.3044	1.3012
$\tau = 9$	1.3170	0.4321	2.9248	0.8771
$\tau = 10$	1.3634	**0.4173**	3.0952	1.3501
$\tau = 11$	1.2963	0.4804	2.8087	0.8381
$\tau = 12$	1.3269	0.4277	2.5917	0.7967
$\tau = 13$	1.2850	0.4395	2.6607	0.7399
$\tau = 14$	1.2558	0.4573	2.6657	0.7263

Table 1: MSE for different values of τ and in recursive prediction.

As it can be seen from the results, the optimal subset of variables selected is given by $\tau = 10$ with $s(t+1)$ as an additional 11th variable. The results show also a strong improvement by considering the time step within the season $s(t+h)$ when performing the recursive long term prediction.

The final test MSE obtained for 50 values using the test dataset is 0.4305 (for the best model without $s(t+1)$ the test MSE was 0.7301).

6.2 Direct prediction

When performing direct long term prediction, the learning methodology is much more complex, since several models have to be trained separately. The models will have the shape

$$\hat{x}(t+h) = F\left(s(t+h), x\left(t-0\right),\ x\left(t-1\right), \ldots\ x\left(t-\tau\right)\right) \tag{8}$$

where h takes the value corresponding to each of the needed prediction horizon.

6.2.1 Variable Selection

Again, the variable selection procedure is performed for different values of τ, evaluating too the inclusion of $s(t+h)$ as an additional variable. The training of the LS-SVMs is performed using grid-search cross-validation (see section 4). The training and test data sets are distributed as mentioned in section 5.

Table 2 shows the training and validation MSE when performing the variable selection procedure for different values of τ, when taking and not taking into

account the time step within the season $s(t+1)$, for the model $\hat{x}(t+1)$ (see equation 8).

τ	with $s(t+1)$ training MSE	with $s(t+1)$ validation MSE	without $s(t+1)$ training MSE	without $s(t+1)$ validation MSE
$\tau = 1$	0.1288	0.1374	0.2054	0.1942
$\tau = 2$	0.1178	0.1323	0.1858	0.1786
$\tau = 3$	0.1143	0.1410	0.1638	0.1811
$\tau = 4$	0.1144	0.1418	0.1471	0.1648
$\tau = 5$	0.1147	0.1412	0.1380	0.1599
$\tau = 6$	0.1228	0.1429	0.1349	0.1565
$\tau = 7$	0.1185	0.1448	0.1209	0.1615
$\tau = 8$	0.1093	0.1546	0.1365	0.1545
$\tau = 9$	0.1036	0.1430	0.1236	0.1451
$\tau = 10$	0.1063	0.1444	0.1208	0.1512
$\tau = 11$	0.1163	**0.1382**	0.1244	0.1442
$\tau = 12$	0.1147	0.1383	0.1352	0.1538
$\tau = 13$	0.1188	0.1431	0.1159	0.1412
$\tau = 14$	0.0981	0.1401	0.1249	0.1495
$\tau = 15$	0.1095	0.1384	0.1196	0.1453

Table 2: MSE for different values of τ for the model $\hat{u}$.

As it can be seen from the results, the optimal subset of variables selected is given by $\tau = 10$ and including $s(t+1)$ as an additional variable.

The process should be similar for each of the model to be trained, $\hat{x}(t+h)$ for $h = 1..50$. However, we consider that the number of simulations needed would be excessive for the given problem. Thus, the same variables selected for $\hat{x}(t+1)$ are used for the rest of the 49 direct models.

The MSE for the 50 direct models in training is 0.63, nevertheless, the test MSE for the 50 direct models is 1.14. This shows the increasing difficulty when modeling long term data dependencies.

6.3 Prediction of the next 50 values of the time series

In this section we present the prediction of the next 50 values of the time series using both approaches. *fig.* 4 shows the recursive prediction and *fig.* 5 shows the direct prediction of the next 50 values of the series. The number of samples in the time series is 875. The goal of the competition is the prediction of the 50 next values of the time series. The evaluation of the performance will be done using the MSE obtained from the prediction of both the 15 and the 50 next values.

In this example, due to the better behaviour of the recursive model in the test dataset, the recursive prediction was selected for the competition.

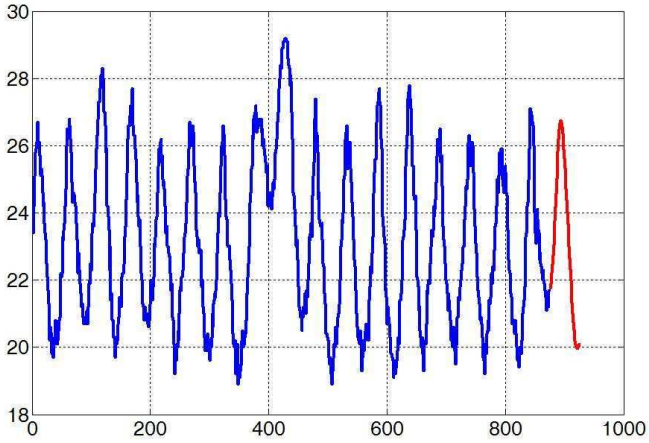

Fig. 4: Recursive prediction of the next 50 values of the time series

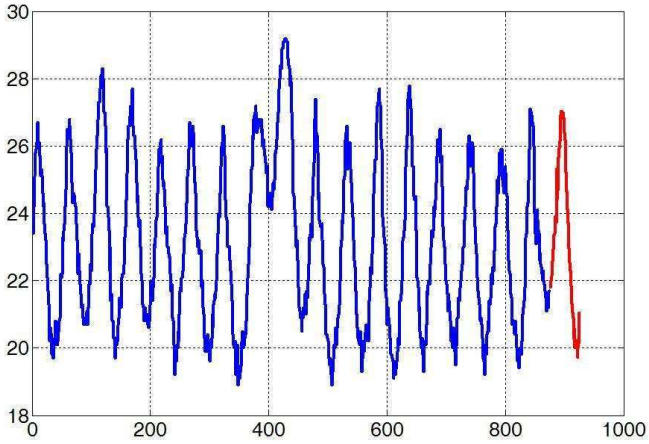

Fig. 5: Direct prediction of the next 50 values of the time series

7 Conclusions

LS-SVMs have been shown to bring excellent performance on time series and function approximation problems in several previous works. However it is important to properly analyze the specific series being dealt with in order to obtain optimal results. When dealing with a time series problem that presents seasonality is is necessary to include the seasonality information in the model. In this paper, a simple approach has be used to include seasonality information on

the LS-SVM model. Both recursive and direct long term time series prediction have been evaluated in the work, using also a simple wrapper variable selection method. The results obtained have shown that when the information of the time step within the season is included in the model, the performance of the predictors increases considerably. The problem considered in this work is the time series forecasting proposed in the ESTSP'2007.

References

[1] Weigend, AS., Gershenfeld, NA.: Time Series Prediction: Forecasting the Future and Understanding the Past, Addison-Wesley (1993)

[2] Yongnan Ji, Jin Hao, Nima Reyhani, Amaury Lendasse: Direct and Recursive Prediction of Time Series Using Mutual Information Selection. IWANN 2005, Lecture Notes in Computer Science. New York: Springer, **3512** (2005) 1010–1017

[3] Lendasse, A., Ji, Y. Reyhani, N., Verleysen, M.: LS-SVM Hyperparameter Selection with a Nonparametric Noise Estimator, ICANN'05, Lecture Notes in Computer Science, Springer **3697** (2005) 625–630.

[4] LS-SVMlab: a MATLAB/C toolbox for Least Squares Support Vector Machines: http://www.esat.kuleuven.ac.be/sista/lssvmlab

[5] Schoelkopf, B., Smola, A.: Learning with Kernels. Cambridge, MA: MIT Press (2002)

[6] Suykens, J.A.K., Van Gestel, T., De Brabanter, J. , De Moor, B., Vandewalle, J.: Least Squares Support Vector Machines, World Scientific, Singapore, (2002)

[7] Jung, T., Herrera, L.J., Schoelkopf, B.: Long Term Prediction of Product Quality in a Glass Manufacturing Process Using a Kernel Based Approach, IWANN 2005, Lecture Notes in Computer Science. New York: Springer, **3512** (2005) 960–967

[8] EUNITE Competition 2003: Prediction of product quality in glass manufacturing, www.eunite.org (2003)

[9] Jarkko Tikka, Jaakko Hollmén, Amaury Lendasse: Input Selection for Long-Term Prediction of Time Series. IWANN 2005: 1002-1009

[10] Hai-shan Wu, Shen Zhang: Power Load Forecasting with Least Squares Support Vector Machines and Chaos Theory, ICNN-B '05, Vol. 2, 1020- 1024

[11] Espinoza M, Joye C, Belmans R, De Moor B (2005) Short term load forecasting, profile identification and customer segmentation: a methodology based on periodic time series. IEEE Trans Power Syst 20(3): 1622-1630

[12] L. J. Herrera, H. Pomares, I. Rojas, A. Guillén, A. Prieto, O. Valenzuela: Recursive Prediction for Long Term Time Series Forecasting Using Advanced Models, Neurocomputing, 2006, Accepted

Time Series Prediction as a Problem of Missing Values

Antti Sorjamaa and Amaury Lendasse *

Helsinki University of Technology - Laboratory of Computer and Information Science
P.O. Box 5400, 02015 HUT - Finland

Abstract. In this paper, time series prediction is considered as a problem of missing values. A new method for the determination of the missing time series values is presented. The new method is based on two projection methods: a nonlinear one (Self-Organized Maps) and a linear one (Empirical Orthogonal Functions). The presented global methodology combines the advantages of both methods to get accurate candidates for prediction values. The methods are applied to a time series competition dataset.

1 Introduction

The presence of missing values in the underlying time series is a recurrent problem when dealing with databases. Number of methods have been developed to solve the problem and fill the missing values. The methods can be classified into two distinct categories: deterministic methods and stochastic methods.

Self-Organizing Maps [1] (SOM) aim to ideally group homogeneous individuals, highlighting a neighborhood structure between classes in a chosen lattice. The SOM algorithm is based on unsupervised learning principle where the training is entirely stochastic, data-driven. No information about the input data is required. Recent approaches propose to take advantage of the homogeneity of the underlying classes for data completion purposes [2]. Furthermore, the SOM algorithm allows projection of high-dimensional data to a low-dimensional grid. Through this projection and focusing on its property of topology preservation, SOM allows nonlinear interpolation for missing values.

Empirical Orthogonal Function (EOF) [3] models are deterministic enabling linear projection to high-dimensional space. They have also been used to develop models for finding missing data [4]. Moreover, EOF models allow continuous interpolation of missing values, but are sensitive to the initialization.

This paper describes a new method, which combines the advantages of both the SOM and the EOF. The nonlinearity property of the SOM is used as a denoising tool and then continuity property of the EOF method is used to recover missing data efficiently.

The SOM is presented in the Section 3, the EOF in Section 4 and the global methodology SOM+EOF in Section 5. Section 6 presents the experimental results using a new competition dataset.

*Part the work of A. Sorjamaa and A. Lendasse is supported by the project of New Information Processing Principles, 44886, of the Academy of Finland. The work of A. Lendasse is supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

2 Time Series Prediction

2.1 Data with Missing Values

In time series prediction problem, the samples are generated by sliding a fixed window over the time series and taking each window full of values as a sample. The size of the window and thus the length of the samples is T. All samples are collected to a *regressor matrix*

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_j \end{bmatrix}, \; j = 1, 2, ..., N, \tag{1}$$

where N is the number of samples and each $\mathbf{x}_j$ is a T-dimensional sample vector.

When predicting the future of the time series, the missing values are added to the end of the known values of the time series. Then, logically the regressor matrix is missing some values in the lower right corner. The shape and the size of the area of the missing values depend on the used method and the horizon of prediction.

2.2 Prediction Strategy

There are three prediction strategies for the long-term prediction of time series that are mainly used. The first and the least calculation intensive is the *Recursive* prediction strategy, where the model selected in the learning phase for the first time step is used repeatedly, or recursively, as far as necessary. The predicted values are used as known values and the prediction is done always only one step at a time.

The next alternative is to use different model to predict each time step. This *Direct* prediction strategy needs different model for each time step and is therefore many times more calculation intensive. In many cases the Direct is still appealing choice, because of the increased accuracy compared to the Recursive strategy. Where the Recursive strategy suffers from accumulation of prediction errors, the Direct does not.

Third alternative is to use a mix of the two, called *DirRec* prediction strategy [5]. With this prediction strategy different model is trained for each time step and all predicted values are used as known values in the process. It means that the regressor is increased by one in every time step when the previous prediction is included in the learning data. This increases the calculation time in the learning process but in many cases, the accuracy is also better.

In this case, when the time series prediction is considered as a missing value problem, the whole set of values to be predicted is estimated at once. Strictly speaking the strategy used here is none of the above, but instead *all-at-once* strategy.

3 Self-Organizing Map

The SOM algorithm is based on an unsupervised learning principle, where training is entirely data-driven and no information about the input data is required [1]. Here we use a 2-dimensional network, compound in c units (or code vectors) shaped as a square *lattice*. Each unit of a network has as many weights as the length T of the learning data samples, $\mathbf{x}_n$, $n = 1, 2, ..., N$. All units of a network can be collected to a weight matrix $\mathbf{m}(t) = [\mathbf{m}_1(t), \mathbf{m}_2(t), ..., \mathbf{m}_c(t)]$ where $\mathbf{m}_i(t)$ is the T-dimensional weight vector of the unit i at time t and t represents the steps of the learning process. Each unit is connected to its neighboring units through neighborhood function $\lambda(\mathbf{m}_i, \mathbf{m}_j, t)$, which defines the shape and the size of the neighborhood at time t. Neighborhood can be constant through the entire learning process or it can change in the course of learning.

Learning starts by initializing the network node weights randomly. Then, for randomly selected sample $\mathbf{x}_{t+1}$, we calculate a Best Matching Unit (BMU), which is the neuron whose weights are closest to the sample. BMU calculation is defined as

$$\mathbf{m}_{BMU(\mathbf{x}_{t+1})} = \arg \min_{\mathbf{m}_i, i \in I} \{||\mathbf{x_{t+1}} - \mathbf{m}_i(t)||\}, \tag{2}$$

where $I = [1, 2, ..., c]$ is the set of network node indices, BMU denotes the index of the best matching node and $||.||$ is standard Euclidean norm.

If the randomly selected sample includes missing values, the BMU cannot be solved outright. Instead, an adapted SOM algorithm, proposed by Cottrell and Letrémy [6], is used. The randomly drawn sample $\mathbf{x_{t+1}}$ having missing value(s) is split into two subsets $\mathbf{x}_{t+1}^T = NM_{\mathbf{x_{t+1}}} \cup M_{\mathbf{x_{t+1}}}$, where $NM_{\mathbf{x_{t+1}}}$ is the subset where the values of $\mathbf{x_{t+1}}$ are not missing and $M_{\mathbf{x_{t+1}}}$ is the subset where the values of $\mathbf{x_{t+1}}$ are missing. We define a norm on the subset $NM_{\mathbf{x_{t+1}}}$ as

$$||\mathbf{x_{t+1}} - \mathbf{m}_i(t)||_{NM_{\mathbf{x_{t+1}}}} = \sum_{k \in NM_{\mathbf{x_{t+1}}}} (\mathbf{x}_{t+1,k} - \mathbf{m}_{i,k}(t))^2, \tag{3}$$

where $\mathbf{x}_{t+1,k}$ for $k = [1, ..., T]$ denotes the k^{th} value of the chosen vector and $\mathbf{m}_{i,k}(t)$ for $k = [1, ..., T]$ and for $i = [1, ..., c]$ is the k^{th} value of the i^{th} code vector.

Then the BMU is calculated with

$$\mathbf{m}_{BMU(\mathbf{x}_{t+1})} = \arg \min_{\mathbf{m}_i, i \in I} \left\{ ||\mathbf{x_{t+1}} - \mathbf{m}_i(t)||_{NM_{\mathbf{x_{t+1}}}} \right\}. \tag{4}$$

When the BMU is found the network weights are updated as

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) - \varepsilon(t) \lambda \left(\mathbf{m}_{BMU(\mathbf{x}_{t+1})}, \mathbf{m}_i, t \right) [\mathbf{m}_i(t) - \mathbf{x_{t+1}}], \forall i \in I, \tag{5}$$

where $\varepsilon(t)$ is the adaptation gain parameter, which is $]0, 1[$-valued, decreasing gradually with time. The number of neurons taken into account during the

weight update depends on the neighborhood function $\lambda(\mathbf{m}_i, \mathbf{m}_j, t)$. The number of neurons, which need the weight update, usually decreases with time.

After the weight update the next sample is randomly drawn from the data matrix and the procedure started again by finding the BMU of the sample. The recursive learning procedure is stopped when the SOM algorithm has converged.

Once the SOM algorithm has converged, we obtain some clusters containing our data. Cottrell and Letrémy proposed to fill the missing values of the dataset by the coordinates of the code vectors of each BMU as natural first candidates for missing value completion:

$$\pi_{(M_{\mathbf{x}})}(\mathbf{x}) = \pi_{(M_{\mathbf{x}})}\left(\mathbf{m}_{BMU(\mathbf{x})}\right), \tag{6}$$

where $\pi_{(M_{\mathbf{x}})}(.)$ replaces the missing values $M_{\mathbf{x}}$ of sample $\mathbf{x}$ with the corresponding values of the BMU of the sample. The replacement is done for every data sample and then the SOM has finished filling the missing values in the data.

The procedure is summarized in Table 1. There is a toolbox available for performing the SOM algorithm in [7].

Table 1: Summary of the SOM algorithm for finding the missing values.

1. SOM node weights are initialized randomly
2. SOM learning process begins
   (a) Input $\mathbf{x}$ is drawn from the learning data set $\mathbf{X}$
      i. If $\mathbf{x}$ does not contain missing values, BMU is found according to Equation 2
      ii. If $\mathbf{x}$ contains missing values, BMU is found according to Equation 4
   (b) Neuron weights are updated according to Equation 6
3. Once the learning process is done, for each observation containing missing values, the weights of the BMU of the observation are substituted for missing values

4 Empirical Orthogonal Functions

This section presents Empirical Orthogonal Functions (EOF) [3]. In this paper, EOF are used as a denoising tool and for finding the missing values at the same time [4].

The EOF are calculated using standard and well-known Singular Value Decomposition (SVD)

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^* = \sum_{k=1}^{K} \rho_k \mathbf{u}_k \mathbf{v}_k, \tag{7}$$

where $\mathbf{X}$ is 2-dimensional data matrix, $\mathbf{U}$ and $\mathbf{V}$ are collections of singular vectors $\mathbf{u}$ and $\mathbf{v}$ in each dimension respectively, $\mathbf{D}$ is a diagonal matrix with the singular values ρ in its diagonal and K is the smaller dimension of $\mathbf{X}$ (or the number of nonzero singular values if $\mathbf{X}$ is not full rank). The singular values and the respective vectors are sorted to decreasing order.

When EOF are used to denoise the data, not all singular values and vectors are used to reconstruct the data matrix. Instead, it is assumed that the vectors corresponding to larger singular values contain more data with respect to the noise than the ones corresponding to smaller values [3]. Therefore, it is logical to select q largest singular values and the corresponding vectors and reconstruct the denoised data matrix using only them.

In the case where $q < K$, the reconstructed data matrix is obviously not the same than the original one. The larger q is selected, the more original data, which also includes more noise, is preserved. The optimal q is selected using validation methods, for example [8].

EOF (or SVD) cannot be directly used with databases including missing values. The missing values must be replaced by some initial values in order to use the EOF. This replacement can be for example the mean value of the whole data matrix $\mathbf{X}$ or the mean in one direction, row wise or column wise. The latter approach is more logical when the data matrix has some temporal or spatial structure in its columns or rows.

After the initial value replacement the EOF process begins by performing the SVD and the selected q singular values and vectors are used to build the reconstruction. In order not to lose **any** information, only the missing values of $\mathbf{X}$ are replaced with the values from the reconstruction. After the replacement, the new data matrix is again broken down to singular values and vectors with the SVD and reconstructed again. The procedure is repeated until convergence criterion is fulfilled.

The procedure is summarized in Table 2.

5 Global Methodology

The two methodologies presented in the previous two sections are combined and the global methodology is presented. The SOM algorithm for missing values is first ran through performing a nonlinear projection for finding the missing values. Then, the result of the SOM estimation is used as initialization for the EOF method. The global methodology is summarized in Table 1

For the SOM we must select the optimal grid size c and for the EOF the optimal number of singular values and vectors q to be used. This is done using validation, using the same validation set for all combinations of the parameters c and q. Finally, the combination of SOM and EOF that gives the smallest validation error is used to perform the final filling of the data.

Table 2: Summary of the EOF method for finding missing values.

1. Initial values are substituted into missing values of the original data matrix $\mathbf{X}$
2. For each q from 1 to K
   (a) SVD algorithm calculates q singular values and eigenvectors
   (b) A number of values and vectors are used to make the reconstruction
   (c) The missing values from the original data are filled with the values from the reconstruction
   (d) If the convergence criterion is fulfilled, the validation error is calculated and saved and the next q value is taken under inspection. If not, then we continue from step a) with the same q value
3. The q with the smallest validation error is selected and used to reconstruct the final filling of the missing values in $\mathbf{X}$

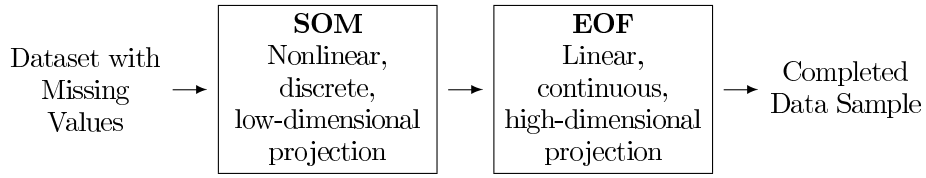

Fig. 1: Global methodology summarized.

Even the SOM as well as the EOF are able to fill the missing values alone, the experimental results demonstrate that together the accuracy is better. The fact that these two algorithms suit well together is not surprising. Two approaches can be considered to understand the complementarity of the algorithms.

Firstly, the SOM algorithm allows nonlinear projection. In this sense, even for dataset with complex and nonlinear structure, the SOM code vectors will succeed to capture the nonlinear characteristics of the inputs. However, the projection is done on a low-dimensional grid (in our case two-dimensional) with the possibility of losing the intrinsic information of the data.

The EOF method is based on a linear transformation using the Singular Value Decomposition. Because of the linearity of the EOF approach, it will fail to reflect the nonlinear structures of the dataset, but the projection space can be as high as the dimension of the input data and remain continuous.

There is a toolbox for performing the SOM+EOF in [9].

6 Experimental Results

In this paper, the ESTSP2007 competition dataset is used as an example. It includes a total of 875 values. The dataset is shown in Figure 2.

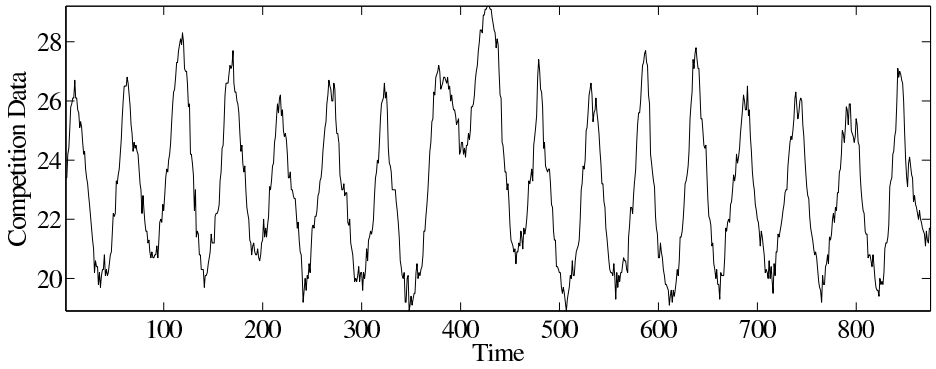

Fig. 2: Competition dataset.

For the SOM algorithm, the dataset is divided into two sets, learning and validation set. The learning set consists of 465 first values and the rest belongs to the validation set. The optimal regressor size is set to 11 after many trial and error experiments.

The optimal SOM size is selected using a simple validation procedure, where the SOM learning is performed using only the learning set and the validation set is used to tune the SOM size for one step ahead prediction. The validation errors are shown in Figure 3.

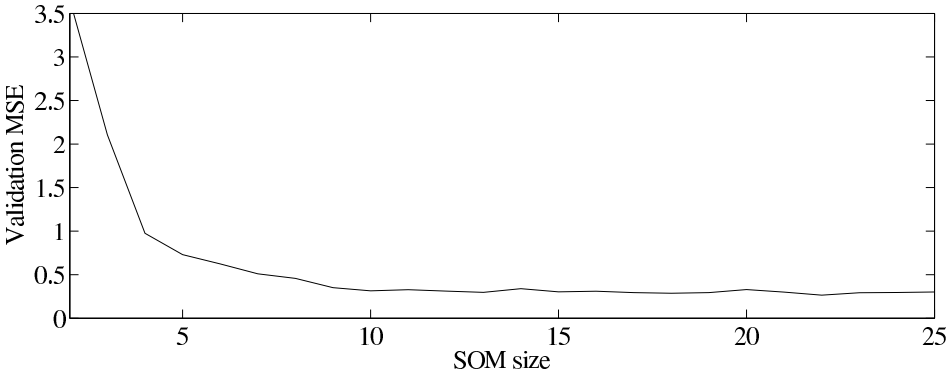

Fig. 3: Validation errors with respect to the SOM grid size.

From Figure 3 the optimal SOM size is selected to 13×13 with validation

error of 0,297. There is only very small difference in the validation error with larger SOM sizes.

The only parameter of the EOF method is tuned using the same learning and validation sets than with the SOM to get comparable results. Also the regressor size is kept the same than with the SOM and the optimization is done for one step ahead prediction. The validation errors are shown in Figure 4.

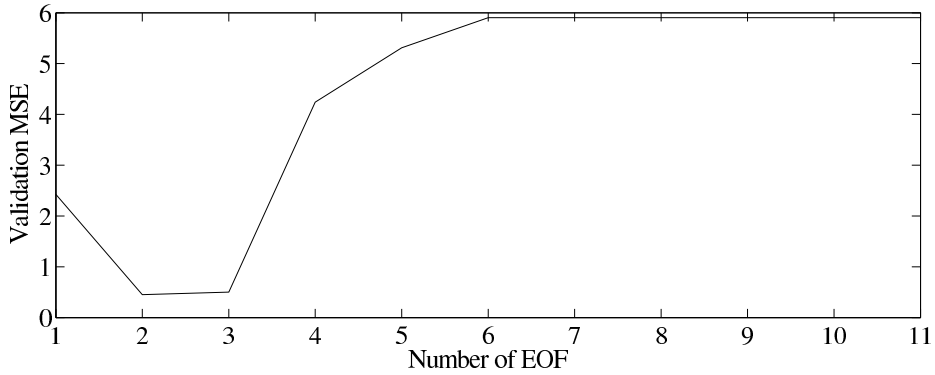

Fig. 4: Validation errors with respect to the number of EOF.

From Figure 4 the optimal number of EOF is selected to 2 with validation error of 0,451. The result suggests relatively strong noise influence in the singular values after the third one, where the validation error is increasing rapidly.

For the SOM+EOF method the two separate methods are combined and the validation is performed for each combination of the SOM sizes and the number of EOF. The validation errors are shown in Figure 5 and 6.

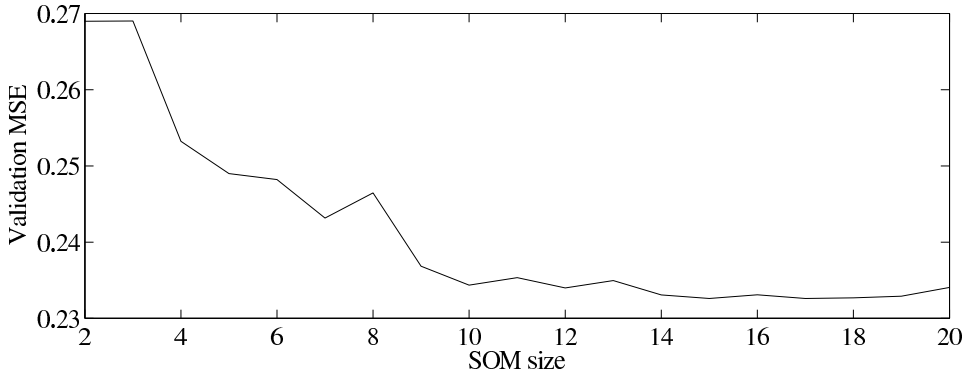

Fig. 5: Minimum validation errors with respect to the SOM size using the SOM+EOF method.

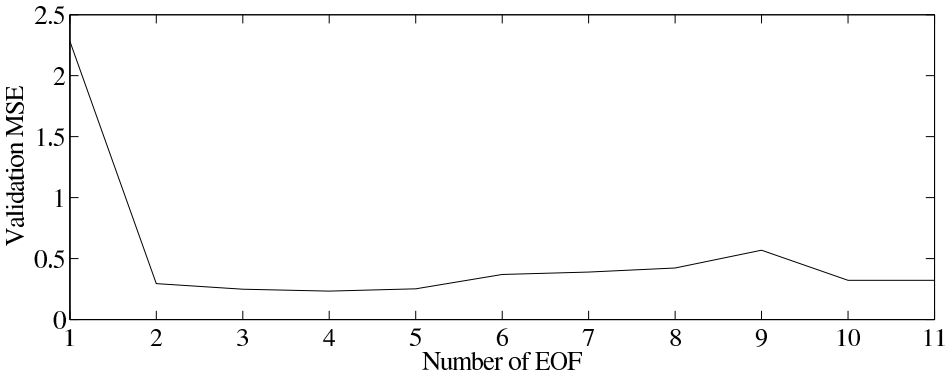

Fig. 6: Validation errors with respect to the number of EOF using SOM size 15×15.

From Figure 5 the optimal SOM is selected to be 15×15 and from Figure 6 the optimal number of EOF to 4 with the validation error of 0,233.

For one step ahead prediction the regressor size is selected to 11, but for the 50 steps ahead the regressor size is increased to 60 in order to fit the missing values to the regressor.

Our experiments with several other datasets have shown that the EOF method uses larger number of EOF when the regressor size is increased. Therefore, the final prediction is done using the number of EOF fixed to 8. The prediction of the 50 timesteps is shown in Figure 7.

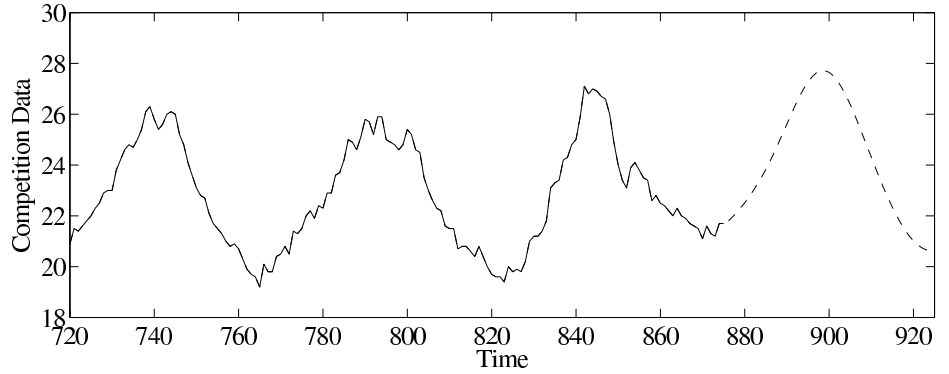

Fig. 7: Prediction of 50 next values of the competition dataset. The real values are presented by the solid line and the dashed one presents the prediction.

From the Figure 7 it seems that that the prediction has removed the noise and is predicting the next peak of the time series quite well.

7 Conclusion

In this paper, we have presented 3 methods for finding missing values in temporal database. The methods are Self-Organizing Maps (SOM), Empirical Orthogonal Function (EOF) and the combination of the two SOM+EOF. The methods are used to find the future values of a time series.

The advantages of the SOM include the ability to perform nonlinear projection of high-dimensional data to lower dimension with interpolation between discrete data points.

For the EOF, the advantages include high-dimensional linear projection of high-dimensional data and the speed and the simplicity of the method.

The SOM+EOF includes the advantages of both individual methods, leading to a new accurate approximation methodology for the missing future values of a time series. The performance obtained show the accuracy of the new methodology.

It is also evident that the EOF is greatly dependent from good initialization in order to produce accurate results. The SOM gives good initialization even the method alone is not so accurate. The two methods complete each other and work well together.

For further work, the modifications and performance upgrades for the global methodology are investigated and applied to other types of datasets from other fields of science, for example climatology and finance.

References

[1] Teuvo Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 1995.

[2] Shouhong Wang. Application of self-organising maps for data mining with incomplete data sets. *Neural Computing and Applications*, 12(1):42–48, 2003.

[3] R. Preisendorfer. *Principal Component Analysis in Meteorology and Oceanography*. Elsevier, 1988.

[4] J. Boyd, E. Kennelly, and P. Pistek. Estimation of eof expansion coefficients from incomplete data. *Deep Sea Research*.

[5] Antti Sorjamaa and Amaury Lendasse. Time series prediction using dirrec strategy. pages 143–148. European Symposium on Artificial Neural Networks, ESANN 2006, Bruges (Belgium), 26-28 April, 2006.

[6] Marie Cottrell and Patrick Letrémy. Missing values: Processing with the kohonen algorithm. pages 489–496. Applied Stochastic Models and Data Analysis, Brest, France, 17-20 May, 2005.

[7] SOM Toolbox: http://www.cis.hut.fi/projects/somtoolbox/.

[8] Amaury Lendasse, V. Wertz, and Michel Verleysen. Model selection with cross-validations and bootstraps - application to time series prediction with rbfn models. In *LNCS*, number 2714, pages 573–580, Berlin, 2003. ICANN/ICONIP (2003), Springer-Verlag.

[9] SOM+EOF Toolbox: http://www.cis.hut.fi/projects/tsp/?page=Downloads.

A neural network model for the prediction of the ESTSP'07 competition data.

Beatriz Pérez-Sánchez and Bertha Guijarro-Berdiñas and Oscar Fontenla-Romero *

Faculty of Informatics - Department of Computer Science
Campus de Elviña 5, 15071, A Coruña- Spain

Abstract.
In this paper a feedforward neural network model is described to tackle with the problem proposed by the First European Symposium on Time Series Prediction. The goal of the competition is to predict, at least, the 50 next values of the given series. The evaluation of the entries' performance will be done using the MSE obtained from the prediction of both the 15 and the 50 next values. In order to obtain our proposed predictive model, different neural networks were trained with the Sensitivity Based Linear Learning method, varying the number of hidden units and using input time windows of 50 or 100 samples. The error rate of every trained network was estimated using a 10-fold cross validation and the obtained results were averaged over 5 different simulations, in order to obtain an almost unbiased estimator of the performance. Finally, the best model was a neural network using 100 inputs, 5 hidden neurons with logistic sigmoid activation functions and 1 linear output neuron. This network obtained a mean MSE test error of 0.774 when predicting 15 samples, and 1.351 when predicting 50 samples.

1 Introduction

This paper presents the application of a neural network model to a time series prediction problem. The work has been carried out in the context of the World-wide competition within the first European Symposium on Time Series Prediction. The competition is announced as an event in the fields of Statistics, Neural Networks, Machine Learning, Control and Econometrics.

This analysis is motivated by the significant advantages that are expected to bring when using more accurate prediction technology than the actual available in fields like finance, sociology, etc. At this respect, a new challenge in the field of time series prediction is the Long-Term Prediction. In this case, from a given past of the time series not only one but several steps ahead have to be predicted.

Given a data set, the aim of the competition is to develop a model capable of predicting the 50 next values (or more) of the time series. Figure 1 shows the 875 samples of the provided time series. No other knowledge about the problem was given. To evaluate the quality of the results two measures will be used: the MSE obtained from the prediction of the 15 and the 50 future values.

In next sections we describe the steps followed to develop the particular proposed model for this competition, as well as the results obtained.

*This work has been partially funded by the project PGIDT05TIC10502PR of the Xunta de Galicia.

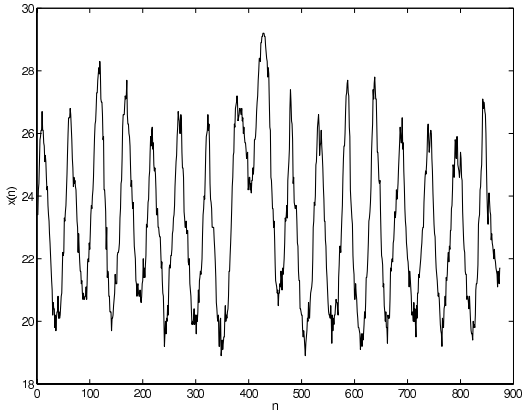

Fig. 1: Time series provided for the competition.

2 Development of the prediction model

The problem of time series prediction can be viewed as a function approximation problem, because a system tries to find the mapping that generates the next instances of the time series based on previous data and desired responses. The aim of function approximation is to estimate a complex function $f(\boldsymbol{x})$ by means of another function $\hat{f}(\boldsymbol{x})$ composed of basis functions,

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i \phi_i(\boldsymbol{x}) \tag{1}$$

where α_i are the coefficients, and $\phi_i(\boldsymbol{x}); i = 1, 2, \cdots, m$ are the basis functions. Many methods have been proposed to select the number and type of these functions and to obtain the corresponding coefficients. In this case, the problem has been approached from the point of view of artificial neural networks (ANN). These are powerful models that have been employed to solve several practical problems, including regression problems, showing an excellent performance [1].

2.1 The Sensitivity-Based Linear Learning algorithm

The learning methods for feedforward neural networks find the network's optimal parameters through a gradient descent mechanism starting from an initial state of these parameters. The first successful algorithm was the classical backpropagation [2]. Although this approach is very useful it has two main drawbacks [3]: the convergence to local minima and a slow learning speed.

In order to solve these problems, several variations of this initial algorithm and also new methods have been proposed (see, for example, [3, 4, 5]). In this work a new sensitivity analysis based learning method (SBLLM) proposed by the authors for two-layer feedforward neural networks has been used [6]. This

method used a linear procedure to obtain the weights of each layer. First, random values are assigned to the outputs of the first layer; later, these initial values are updated based on sensitivity formulas, and finally the weights are calculated using a linear system of equations. This method presents the advantage of achieving a good solution in very few epochs using few computational time.

Consider the two-layer feedforward neural network in Figure 2 where S the number of data samples, I is the number of inputs x_{is}, J the number of outputs y_{js}, K the number of hidden units with outputs z_{ks}, $x_{0s} = 1$, $z_{0s} = 1$, w are the weights and f the activation functions.

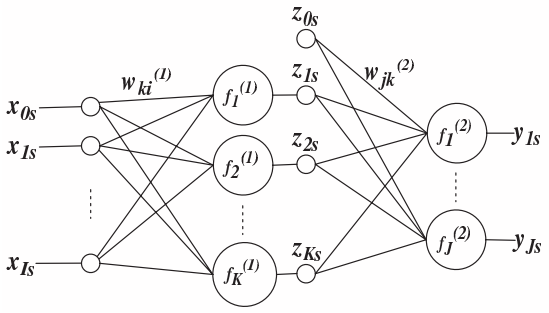

Fig. 2: Two-layer feedforward neural network.

The superscripts (1) and (2) are used to refer to the first and second layer. This network can be considered to be composed of two one-layer neural networks. Usually, weights are updated using the mean squared error (MSE) as cost or error function. This function estimates the error of the system comparing the real and the desired output. In this work, assuming that the intermediate layer outputs $\mathbf{z}$ are known, a new cost function for this network is defined as:

$$\begin{aligned} Q(\mathbf{z}) &= Q^{(1)}(\mathbf{z}) + Q^{(2)}(\mathbf{z}) = \\ &= \sum_{s=1}^{S} \left[\sum_{k=1}^{K} \left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - f_k^{(1)^{-1}}(z_{ks}) \right)^2 + \right. \\ &\quad \left. \sum_{j=1}^{J} \left(\sum_{k=0}^{K} w_{jk}^{(2)} z_{ks} - f_j^{(2)^{-1}}(y_{js}) \right)^2 \right]. \end{aligned}$$

This cost function is based on the sum of squared errors obtained, independently, by the hidden and output layers.

In the proposed algorithm, after initializing the z_{ks} to small random values,

the sensitivities of the cost function with respect to z_{ks} are calculated as:

$$\begin{aligned}\frac{\partial Q}{\partial z_{ks}} &= \frac{\partial Q^{(1)}}{\partial z_{ks}} + \frac{\partial Q^{(2)}}{\partial z_{ks}} = \\ &= -\frac{2\left(\sum_{i=0}^{I} w_{ki}^{(1)} x_{is} - {f_k^{(1)}}^{-1}(z_{ks})\right)}{f_k^{'(1)}(z_{ks})} + \\ & 2\sum_{j=1}^{J}\left(\sum_{r=0}^{K} w_{jr}^{(2)} z_{rs} - {f_j^{(2)}}^{-1}(y_{js})\right) w_{jk}^{(2)}.\end{aligned}$$

with $k = 1, \ldots, K$, as $z_{0s} = 1, \forall s = 1, \ldots, S$.

Next, the values of the intermediate outputs $\mathbf{z}$ are modified using the Taylor series approximation:

$$Q(\mathbf{z} + \Delta\mathbf{z}) = Q(\mathbf{z}) + \sum_{k=1}^{K}\sum_{s=1}^{S} \frac{\partial Q(\mathbf{z})}{\partial z_{ks}} \Delta z_{ks} \approx 0,$$

which leads to the following increment

$$\Delta\mathbf{z} = -\frac{Q(\mathbf{z})}{||\nabla Q||^2} \nabla Q,$$

which is used to update the z_{ks} values. Thus, using the outputs of the intermediate layer z_{ks} we can learn, for each layer independently, the optimal weights $w_{ki}^{(1)}$ and $w_{jk}^{(2)}$ by solving a linear system of equations, as proposed in [7]. As stated in [6], this algorithm offers an interesting combination of speed, reliability and simplicity.

2.2 Preprocessing stage

Dynamic modeling implies a two step procedure [8]. The first step is to transform the observed time series into a trajectory in a reconstruction space by using an embedding technique [9]. The most common is a time delay embedding, which can be implemented with a delay line with a size specified by Takens' embedding theorem. The second step is to build the predictive model from the trajectory in the reconstruction space.

Takens [10] proved that an observable sample $x(n)$ of a dynamic system and its delayed versions $\boldsymbol{x}(n) = [x(n), x(n-\tau), \ldots, x(n-(N-1)\tau)]$, where τ is a specific time delay, can be used to create a trajectory in a Euclidean space of size N, which preserves the dynamical invariants of the original dynamical system. The dimension N of the space must be larger than $2D$, where D is the dimension of the attractor.

Therefore, in order to feed the signal to the ANN, an embedding layer, implemented by a time delay line, was used to transform the original space of the provided time series $x(n)$ into a reconstruction space whose output is a sequence

of N-dimensional state vectors, $\boldsymbol{x}(n) = [x(n), x(n-\tau), \cdots, x(n-(N-1)\tau)]^T$. Specifically, in this case a value of $\tau = 1$ was employed. Moreover, from a visual analysis of the original signal (see Figure 1) it can be observed that it presents a sinusoidal shape with a period composed of around 50 samples. Taking into account the periodic characteristics of the signal, values corresponding to one period ($N = 50$) and two periods ($N = 100$) were evaluated for the embedding dimension N.

Although the objective of the competition was long-term prediction the ANN was initially trained to predict the $x(n+1)$ sample. Later on, the trained network was used to predict up to 50 future values by, recursively, presenting its own outputs as new inputs. So, using $N = 50$, and taking into account the two problems proposed for the competition, 811 different time windows or training patterns were available in order to predict the next 15 samples, while 776 training patterns were available for the prediction of the next 50 samples. These sizes were reduced to 761 and 726, respectively when $N = 100$ was used. Finally, in order to train the network, the signal was previously normalized to have zero mean and a standard deviation of 1.

2.3 Procedure to evaluate the performance of the network

To check the error rate of the proposed artificial neural network and due to the small size of the training set, a resampling technique known as 10-fold crossvalidation was employed [11], as this technique allows to obtain an almost unbiased estimator of the true error rate of a model. Thus, the training set was split into 10 mutually exclusive data sets. Following this method, the network was trained 10 times, selecting each time a different test set with the remaining sets being used for training. Also, this process was repeated 5 times to consider the variability of the results due to the different network's initial z_{ks}. Finally, performance measures were calculated using all these results. Therefore we got 5×10 measures both for training and testing.

These measures are the *Mean Squared Error* (MSE) obtained from the prediction of both the 15 next values and the 50 next values, as required from the competition announcement, i.e.,

$$MSE_1 = \frac{1}{15}\sum_{i=1}^{15}(y_i - \hat{y}_i)^2$$

and

$$MSE_2 = \frac{1}{50}\sum_{i=1}^{50}(y_i - \hat{y}_i)^2.$$

where $\hat{y}_i$ is the neural network's output and y_i is the expected output.

Also, we used a second performance magnitude, the *Maximal Error* (MaxErr), which is defined as:

$$MaxErr_1 = max(|y_i - \hat{y}_i|), i = 1, \cdots, 15$$

and

$$MaxErr_2 = max(|y_i - \hat{y}_i|), i = 1, \cdots, 50.$$

3 Results

Following the previous steps, several neural networks were trained. In every case, the employed activation functions were the logistic sigmoid for hidden neurons and the linear function for the output neuron. Therefore the initial values of the z_{ks} were assigned to an uniformly distributed random variable on the interval $[0.05, 0.95]$. As already mentioned, two dimensions were tried for the input layer: 50 and 100 inputs. The size of the hidden layer were varied between 5 and 30 hidden neurons.

For every 5 simulations, and every test example of the k-fold the measures MSE_1, MSE_2, $MaxErr_1$ and $MaxErr_2$ were calculated and averaged. The best results were obtained using 100 inputs and 5 hidden neurons. In this case, the mean of these measures were $MSE_1 = 0.774$, $MSE_2 = 1.351$, $MaxErr_1 = 3.90$ and $MaxErr_2 = 5.439$. Also, in Figure 3 is shown the evolution of the MSE, as a function of the index i of the sample y_i to be predicted. As can expected, the MSE degrades as the predicted sample leaves the last real available sample.

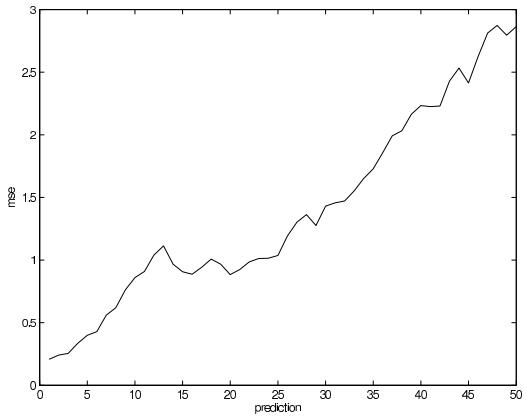

Fig. 3: Evolution of the MSE as a function of the prediction interval.

Figures 4, 5 and 6 show, for the test data, a plot of the real and the desired output for the prediction of the 1st, 15th and 50th future samples, respectively. As can be seen, the highest errors occurs around the interval from the 200th to the 300th samples, which corresponds to the more atypical part of the signal.

Also, Figure 7 plots the real time series versus the generated time series.

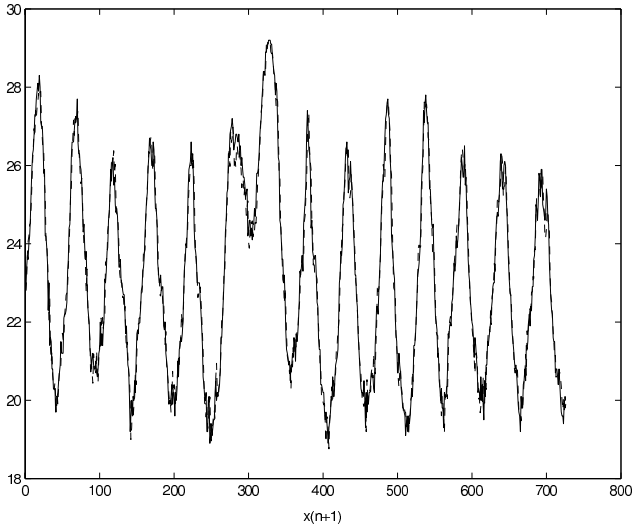

Fig. 4: Observed (solid line) and predicted time series (dashed line) when predicting the $x(n+1)$ sample.

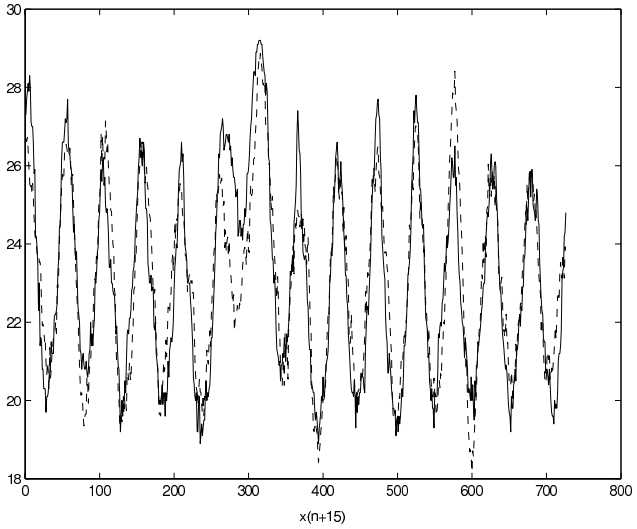

Fig. 5: Observed (solid line) and predicted time series (dashed line) when predicting the $x(n+15)$ sample.

Finally, Table 1 contains the 50 samples predicted by our model from the given time series.

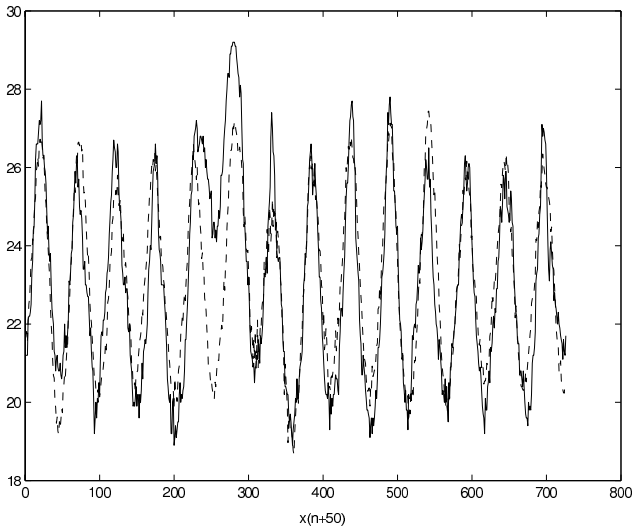

Fig. 6: Observed (solid line) and predicted time series (dashed line) when predicting the $x(n+50)$ sample.

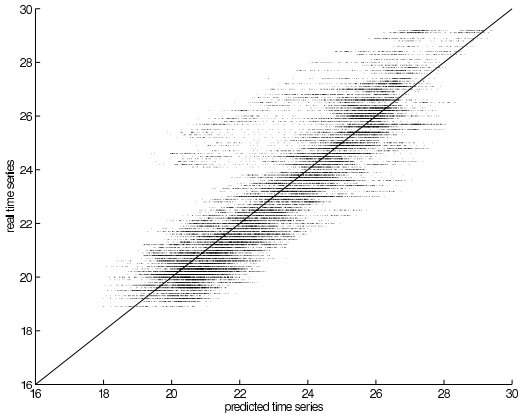

Fig. 7: Observed versus predicted time series.

4 Citation

References

[1] T. Masters. *Signal and image processing with neural networks*. John Wiley & Sons, 1994.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Willian. Learning representations of backpropagation errors. *Nature*, 323:533–536, 1986.

[3] Y. LeCun, L. Bottou, G.B. Orr, and K.-R. Müller. Efficient backprop. In G. B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the trade*, number 1524 in LNCS.

i	$\hat{y}_i$	i	$\hat{y}_i$	i	$\hat{y}_i$
1	19.909	2	20.249	3	20.586
4	20.696	5	21.018	6	21.552
7	21.881	8	22.097	9	22.589
10	22.879	11	22.961	12	23.174
13	23.497	14	23.656	15	24.011
16	24.464	17	24.699	18	24.880
19	24.966	20	24.822	21	24.844
22	24.797	23	24.923	24	24.825
25	24.776	26	24.688	27	24.482
28	24.132	29	23.734	30	23.638
31	23.314	32	23.151	33	22.888
34	22.738	35	22.330	36	21.947
37	21.734	38	21.551	39	21.235
40	21.042	41	21.023	42	20.824
43	20.592	44	20.418	45	20.361
46	20.219	47	20.242	48	20.424
49	20.457	50	20.475		

Table 1: Predictions made from the competition given signal.

Springer-Verlag, 1998.

[4] M. T. Hagan and M. Menhaj. Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993, 1994.

[5] M. F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.

[6] E. Castillo, B. Guijarro-Berdiñas, O. Fontenla-Romero, and A. Alonso-Betanzos. A very fast learning method for neural networks based on sensitivity analysis. *Journal of Machine Learning Research*, 7(Jul):1159–1182, 2006.

[7] E. Castillo, O. Fontenla-Romero, A. Alonso Betanzos, and B. Guijarro-Berdiñas. A global optimum approach for one-layer neural networks. *Neural Computation*, 14(6):1429–1449, 2002.

[8] S. Haykin and J. Principe. Dynamic modeling with neural networks. *IEEE Signal Processing Magazine*, 15, 1988.

[9] A.S. Weigend and N.A. Gershenfeld. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, MA,, 1994.

[10] F. Takens. *Detecting strange attractors in turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381. 1981.

[11] S.H. Weiss and C.A. Kulikowski. *Computer systems that learn. Classification and prediction methods from statistics, neural nets, machine learning and experts systems*. Morgan Kauffman Publishers Inc., San Francisco, 1990.

Meta-learning of Recurrent Neural Networks for Time Series Prediction

M.P. Cuéllar[1], M. Delgado and M.C. Pegalajar

University of Granada - Dept of Computer Science and Artificial Intelligence
E.T.S Ingeniería Informática. 18071. Granada. - Spain.

Abstract. This article describes a meta-learning procedure to solve the vanishing gradient problem in the training of recurrent neural networks. The main idea is to make a global search with an heuristic procedure in order to locate the best areas in the solution space. Then, a non-linear programming algorithm is used to improve the best networks with local training. In the experimental section, we apply the method to solve the Time Series prediction problem proposed in the Competition of ESTSP2007.

1 Introduction

1.1 Motivation

Neural Networks [1] are bio-inspired mathematical tools that have been widely used to solve complex engineering problems, including time series prediction. Many network models have been applied to solve time series tasks, for example feedforward networks, recurrent networks [2][3], autoassociative maps [4], etc. Dynamical Recurrent Neural Networks (DRNNs) [5] are models that may be obtained from a feedforward one, by adding recurrent connections to the network topology. This provides the network with long and short term memory, and makes it suitable to learn data sequences with temporal properties. These network models are also able to learn automatically the data dependencies (supposed to be unknown previously) during the training. Due to these properties, in this work we use DRNNs to solve the time series prediction problem proposed.

However, the main limitations of DRNNs are related to the classic training algorithms. These methods may suffer of the vanishing gradient problems [6], therefore providing unsuitable or local solutions. In the last decade, several solutions have been proposed to avoid this problem, and most of them include heuristic procedures (tabu search, simulated annealing, evolutionary optimization, hybrid algorithms, etc.), for example in [7][8][9]. The main idea of the heuristic approaches is to make a global search in the solution space of the networks, to avoid the local learning provided by the traditional gradient-based training methods. On the other hand, the hybrid methods combine the global search of the heuristic procedures with the local learning of the classic training methods, in order to improve the network performance in a local area of the solution space.

In this work, the meta-learning Scatter-Search procedure [10][7] is used to train DRNNs. This document is structured as follows: Section 2 introduces the DRNN models used for the experiments. Section 3 explains the algorithm

Scatter Search. After that, Section 4 describes how to use the Meta-Learning algorithm to train and optimize the topology of a DRNN. Section 5 shows the experimental results and finally, Section 6 concludes.

2 Dynamical Recurrent Neural Networks

Dynamical Recurrent Neural Networks are input/output mapping models that may be obtained from a feedforward network by adding recurrent connections to the network topology. The DRNN models most known are the Fully Connected Recurrent Neural Network, the Jordan Network [5], and the Elman Network [11]. In this work we use the Elman network, since it has obtained the best results in a previous experimental study. It contains three neuron layers: input data layer, hidden (or processing) layer and output data layer. The recurrent connections are in the hidden layer, so that the output value of a hidden neuron at time t is also input for all hidden neurons at time $t+1$. This provides the Elman network with long and short term memory in time, codified in the network structure by mean of recurrent neurons.

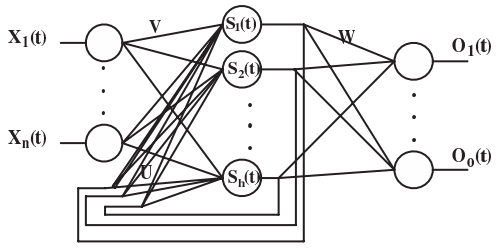

Fig. 1: Example of an Elman Network with n inputs, h hidden neurons, and o outputs.

The diagram in Figure 1 illustrates an example of an Elman recurrent neural network where:

- n, h, o are the number of input, hidden and output neurons, respectively;
- $X_i(t)$ is the input data to neuron i at time t $(1 <= i <= n)$;
- $S_j(t)$ is the output of hidden neuron j at time t $(1 <= j <= h)$;
- $O_k(t)$ is the k-th network output at time t $(1 <= k <= o)$;
- the values U, V, W are matrices that encode the network weights, so that V_{ji} is the weight for the connection from input neuron i to hidden neuron j; U_{jr} is the weight for the recurrent connection from hidden neuron r to hidden neuron j; and W_{kj} is the weight for the connection from hidden neuron j to output neuron k.

The equations for the network dynamics are:

$$S_j(t) = f(\sum_{r=1}^{h} U_{jr} S_r(t-1) + \sum_{i=1}^{n} V_{ji} X_i(t)) \quad (1)$$

$$O_k(t) = g(\sum_{j=1}^{h} W_{kj} S_h(t-1)) \quad (2)$$

In equations 1 and 2, *f(x)* is the sigmoid function and *g(x)* is the identity function.

3 The algorithm Scatter Search

The algorithm *Scatter Search* [7] is a meta-heuristic procedure that combines an heuristic global search with a solution improvement method for a local search in the best areas of the solution space. The algorithm 1 describes the method in depth, where:

- N is the number of initial solutions.
- P is a set of N solutions.
- R is the reference set of solutions. The size of R is b. $R(i)$ denotes the i-th solution in the set R.
- K is the number of subsets of solutions to be generated from the set R.
- R'_i is a subset generated from R.
- H_i is a set of solutions generated by the recombination operator over the set R'_i. The size of H_i is $|H_i|$. $H_i(j)$ denotes the j-th solution in the set H_i.
- The function $f(s)$ is the objective function, to be minimized.
- The procedure $Initialize(N)$ generates a set of N solutions randomly.
- The procedure $BuildReferenceSet(P, b)$ generates the reference set from P, with size b.
- The procedure $sort(R)$ sorts R in increasing order, where the first solution $R(1)$ is the best one $(f(R(1)) \leq f(R(i)), \forall i \in \mathbb{N} : 1 < i \leq b)$.
- The procedure $GenerateSubSet(R)$ returns a subset of solutions in R.
- The procedure $Combine(R_i)$ returns a subset of solutions generated by combining the solutions in R_i.
- The procedure $ImproveSolution(S)$ returns the solution S improved by mean of local optimization.

- The procedure *ReplaceSolution*$(R(i), S)$ replaces the solution $R(i)$ in R with the solution S.

The relevant procedures of above are explained in depth in section 4. However, the reader may obtain further information about the method *Scatter Search* and the main basis of Meta-learning in [7][10].

Algorithm 1 Scatter Search main procedure

```
1: set P= Initialize(N)
2: R= BuildReferenceSet(P, b)
3: Evaluate(R)
4: Sort(R) (the first solution is the best one)
5: while stopping condition is not satisfied do
6:    for i = 1 to K do
7:       R'_i = GenerateSubSet(R);
8:    end for
9:    for i = 1 to K do
10:      H_i = Combine(R'_i);
11:      for j = 1 to |H_i| do
12:         S = H_i(j)
13:         S' = ImproveSolution(S);
14:         if f(S') < f(R(b)) then
15:            ReplaceSolution(R(b), S')
16:            Sort(R) (the first solution is the best one)
17:         end if
18:      end for
19:   end for
20: end while
21: return R
```

4 Meta-Learning for dynamical recurrent neural networks with Scatter Search

This section explains the use of *Scatter Search* to train DRNNs. Firstly, Subsection 4.1 defines the codification of the Elman network into a solution. After that, subsection 4.2 exposes the objective function. Finally, Subsections 4.3-4.6 explain the relevant procedures of *Scatter Search* in Section 3.

4.1 The codification

A solution vector in the *Scatter Search* procedure encodes an Elman network in the following way: Each network connection is assigned to a component in the vector. The value of such component is the value of the network weight for that connection. Thus, the size of a vector solution S encoding an Elman network with n inputs, h hidden neurons and o outputs is:

$$|S| = h(n + h + o) \tag{3}$$

Figure 2 prints and example of the codification of an Elman network with one input, two hidden neurons and one output.

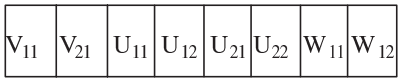

Fig. 2: Example of the codification of an Elman network.

4.2 The objective function

The objective function address the search in the solution space, to find the best network weights for the problem to solve. In this work, the objective is to minimize the error between the network outputs and the desired outputs in the training data set. Equation 4 shows this idea:

$$f(S^*) = min\{f(S)\} = min\{\frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{o}(O_k(t) - D_k(t))^2\} \tag{4}$$

In equation 4, S is a solution vector encoding and Elman network, T is the number of training input/output patterns for the network, and $D_k(t)$ is the desired value for output neuron k at pattern t.

4.3 The Reference Set generation procedure

The reference set R in the *Scatter Search* algorithm has a fixed size b, where b is a parameter for the algorithm. R is built from a larger set P by selecting the $b/2$ best solutions and the $b/2$ most diverse solutions in P. This procedure is explained in depth in algorithm 2

Algorithm 2 Scatter Search Reference Set generation procedure

1: set $R = \emptyset$
2: sort(P) (the first solution is the best one)
3: **for** $i = 1$ to $b/2$ **do**
4: set $R = R \cup \{P(i)\}$
5: **end for**
6: **while** $|R| < b$ **do**
7: set $R = R \cup \{s \in P / \neg\exists s' \in P : ||s' - R(k)|| > ||s - R(k)||, \forall k \in \mathbb{N}, 1 \leq k < |R|\}$
8: **end while**
9: return R

4.4 The SubSet generation procedure

The subset generation procedure select two or more solutions from the reference set R to be combined. In this work, the subset generation procedure returns a set with two solutions obtained with the binary tournament selection operator. The algorithm 3 shows this idea in depth.

Algorithm 3 Scatter Search SubSet generation procedure

1: set $R' = \emptyset$
2: **for** $i = 1$ to 2 **do**
3: set S_1 = choose a solution in R randomly
4: set S_2 = choose a solution (different of S_1) in R randomly
5: **if** $f(S_1) < f(S_2)$ **then**
6: set $R' = R' \cup \{S_1\}$
7: **else**
8: set $R' = R' \cup \{S_2\}$
9: **end if**
10: **end for**
11: return R'

4.5 The Combination procedure

The procedure *Combine(R')* combine the solutions in R' to generate a set of new solutions H. In this work, the size of set $|R'| = 2$, because of the subset generation method studied in section 4.4. Both solutions are combined to generate two new solutions with the $BLX - \alpha$ combination operator. The algorithm 4 explains the combination procedure, where *rand(x,y)* is a method that provides a random real number in $[x, y]$.

Algorithm 4 Scatter Search Combination procedure

1: set $H = \emptyset$
2: **for** $i = 1$ to 2 **do**
3: set $h(i)$= generate new solution vector
4: **for** $j = 1$ to $|R'(1)|$ **do**
5: set $M = Max\{R'_j(1), R'_j(2)\}$, $m = Min\{R'_j(1), R'_j(2)\}$
6: set $h_j(i) = rand(m - (M - m)\alpha, M + (M - m)\alpha)$
7: **end for**
8: set $H = H \cup \{h(i)\}$
9: **end for**
10: return H

4.6 The Improvement procedure

The improvement procedure makes a local search in the environment of a solution to improve its performance. In this work we use a Quasi-Newton method

based on the BFGS formula [12], since it has provided the best results in the experiments about the time series competition data. A solution S is improved iteratively with the BFGS algorithm, depending on a search address d_k at iteration k, and a step length α_k:

$$S_{k+1} = s_k + d_k \alpha_k \tag{5}$$

$$d_k = H_k G_k \tag{6}$$

$$H_{k+1} = H_k + \frac{(S_{k+1} - S_k)(S_{k+1} - S_k)^t}{(S_{k+1} - S_k)^t (G_{k+1} - G_k)} - \frac{H_k (G_{k+1} - G_k)(G_{k+1} - G_k)^t H_k}{(G_{k+1} - G_k)^t H_k (G_{k+1} - G_k)} \tag{7}$$

$$G_k = (\frac{\partial}{\partial V_{11}} f(S_k), ..., \frac{\partial}{\partial V_{hn}} f(S_k), \frac{\partial}{\partial U_{11}} f(S_k), ..., \frac{\partial}{\partial U_{hh}} f(S_k), \frac{\partial}{\partial W_{11}} f(S_k), ..., \frac{\partial}{\partial W_{oh}} f(S_k)) \tag{8}$$

$$\alpha_k = argmin_{\alpha > 0} \{f(S_k + \alpha_{k-1} d_k)\} \tag{9}$$

The reader may find further information about the BFGS algorithm and its application to train recurrent neural networks in [13][9][14][15].

5 The experiments

This section shows the experiments to train an Elman network with the *Scatter Search* meta-learning procedure. Subsection 5.1 introduces the data set: The time series prediction problem of the competition in ESTSP-2007. After that, Subsection 5.2 exposes the parameters. Finally, Subsection 5.3 shows the experimental results and the predicted data.

5.1 The data set: ESTSP-2007 time series competition

The data set is the time series proposed for the competition in ESTSP-2007. It is a set of 875 real values, for which the following 50 ones must be predicted. Figure 3 shows the data of the time series.

5.2 The parameters

Let's assume the following structure for an Elman Network to be trained in the time series prediction data of the competition in ESTSP-2007:

- Input neurons: *1, corresponding to the value $Y(t)$ of the time series.*
- Output neurons: *1, corresponding to the value $Y(t+1)$ to be predicted.*

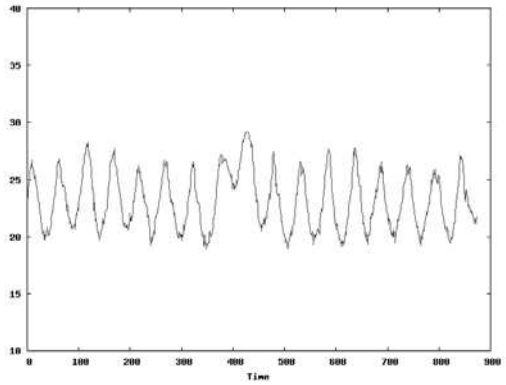

Fig. 3: Data provided for the competition in ESTSP-2007.

- Hidden neurons: *12*.
- Training data set: *first 800 values of the time series*.
- Test data set: *last 75 values of the time series*.

For the prediction, the network output at time t, which is an approximation of $Y(t+1)$, is used as network input at time $t+1$. This procedure is looped for K times until we obtain the value $Y(t+K)$ to be predicted.

The parameters for the *Scatter Search* procedure are:

- Stopping criterion: *50000 solutions evaluated*.
- Size of beginning population P: *500*.
- Size of reference set R: *50*.
- Iterations for the improvement procedure: *30*.
- Value α (for the combination method): *0.5*.

The meta-learning proposal has been run for 50 times with the parameters of above. The following subsection shows the results of the experiments.

5.3 The experimental results

The table 1 shows the Mean Square Error (MSE) of the best, worst and average solutions in the training and test data sets (Columns 1 and 2). It also prints the standard deviation of the MSE (Columns 3 and 4) and the average time of a running of the algorithm (Column 5).

Table 2 prints the 50 values to be predicted in the competition with the best solution, and Figure 4 plots the data learned and predicted.

Type	MSE (tr.)	MSE (test)	s.d. (tr.)	s.d. (test)	Time (secs.)
Average	0.169	1.226	0.015	0.526	651.1
Best	0.160	0.534	–	–	658
Worst	0.176	2.867	–	–	677

Table 1: MSE, s.d. and computational time of the average, best and worst experiments.

876-885	886-895	896-905	906-915	916-925
21,23	22,79	25,98	24,05	20,93
21,02	23,03	25,89	23,85	20,61
21,04	23,42	25,77	23,48	20,47
21,24	23,82	25,63	23,02	20,25
21,38	24,50	25,47	22,83	19,92
21,50	24,66	25,27	22,59	19,76
21,62	24,85	25,02	22,11	19,79
21,79	25,21	24,72	21,60	19,85
22,09	25,72	24,43	21,39	19,92
22,52	25,94	24,21	21,21	20,00

Table 2: Values of the prediction for the next 50 values.

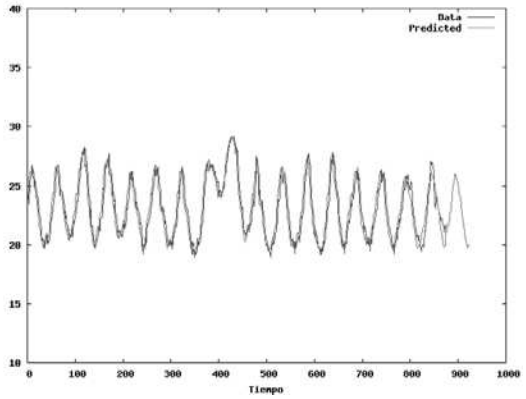

Fig. 4: Real data and prediction by the best solution.

We attempt to check the robustness of the algorithm statistically. The Kolmogorov-Smirnoff normality test has been applied over the MSE in the training and test sets, with 0.05 of confidence level. The *p-value* obtained in the training MSE is 0.2784, while in the test MSE is 0.4435. Neither the traning nor the test results follow a normal distribution. Thus, in an only running of the algorithm we cannot ensure that the solution provided will be suitable. We need to make several runnings of the algorithm to obtain a competitive result. However, this situation may be because of the complexity of the problem to be

solved, if the solution space has a large number of local optimal solutions.

6 Conclusions

In this work, we have developed a meta-learning procedure for dynamical recurrent neural networks, based on the *Scatter Search technique*. It has been applied to the time series prediction data of the competition in ESTSP-2007. The results in the test experiments have provided a suitable prediction for the last 75 values of the known data. However, a large number of local optimal solutions in the problem have not allowed the algorithm to find the best solutions in all the cases, therefore decreasing the robustness of the algorithm to solve this problem.

References

[1] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[2] A. Aussem. Dinamical recurrent neural networks towards prediction and modeling of dynamical systems. *Neurocomputing*, 28(15):207–232, 1999.

[3] X. Cai, N. Zhang, G. Venayagamoorthy, and D. Wunsch. Time series prediction with recurrent neural networks using a hybrid pso-ea algorithm. In *Proc. International Joint Conference on Neural Networks (IJCNN'04)*, Budapest (Hungary), 2004.

[4] A. Lendasse, M. Verleysen, E. De Bodt, M. Cottrel, and P. GrÃ©gorie. Forecasting time series by kohonen classification. In *Proc. European Symposium on Artificial neural Networks*, pages 221–226, 1998.

[5] D.P. Mandic and J.A. Chambers. *Recurrent Neural Networks for Prediction*. John Wiley and sons, 2001.

[6] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. on Neural Networks*, 5(2):157–166, 1994.

[7] M. Laguna and R. Martí. *Scatter Search. Methodology and Implementations in C*. Kluwer Academic Publishers, 2003.

[8] R.B.C. Prudencio and T.B. Ludermir. Neural network hybrid learning: Genetic algorithms and levenberg-marquardt. In *Proc. 26th Annual Conference of the Gesellschaft fÃ¼r Classifikation*, pages 464–472, 2003.

[9] M.P. Cuéllar, M. Delgado, and M.C. Pegalajar. Memetic evolutionary training for recurrent neural networks: an application to time-series prediction. *Expert Systems*, 23(2):99–117, 2006.

[10] R. Martí, M. Laguna, and F. Glover. Principles of scatter search. *European Journal of Operational Research*, 169:359–372, 2000.

[11] J.L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[12] K. Schittkowski and Ch. Zillober. Non-linear programming. Technical Report D-95440, Dept. of mathematics, University of Bayreuth, Bayreuth (Germany), 1955.

[13] R.H. Byrd, P. Lu, and J. Nocedal. A limited memory algorithm for bound constrained optimizetion. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208, 1995.

[14] M.P. Cuéllar, M. Delgado, and M.C. Pegalajar. An application of non-linear programming to train recurrent neural networks in time series prediction problems. In *Proc. of International Conference on Enterprise and Information Systems (ICEIS'05)*, pages 35–42, Miami (USA), 2005.

[15] C. Zhu, R.H. Byrd, and J. Nocedal. L-bfgs-b. algoritmo 778: L-bfgs-b, fortran routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.

Input variable selection for time series prediction with neural networks– an evaluation of visual, autocorrelation and spectral analysis for varying seasonality

Sven F. Crone[1] and Nikolaos Kourentzes[1]

1- Lancaster University Management School, Dept. of Management Science
Lancaster, LA1 4YX, United Kingdom

Abstract. The identification and selection of adequate input variables and lag structures without domain knowledge represents one the core challenges in modeling neural networks for time series prediction. Although a number of linear methods have been established in statistics and engineering, they provide limited insights for nonlinear patterns and time series without equidistant observations and shifting seasonal patterns of varying length, leading to model misspecification. This paper describes a heuristic process and stepwise refinement of competing approaches for model identification for multilayer perceptrons in predicting the ESTSP'07 forecasting competition time series.

1. Introduction

Artificial neural networks (NN) have found increasing consideration in forecasting theory, leading to successful applications in time series prediction and explanatory forecasting [1]. Despite their theoretical capabilities, NN have not been able to confirm their potential in forecasting competitions against established statistical methods, such as ARIMA or Exponential Smoothing [2]. As NN offer many degrees of freedom in the modelling process, from the selection of activation functions, adequate network topologies of input, hidden and output nodes, learning algorithms etc. their valid and reliable use is often considered as much an art as science. Previous research indicates, that the parsimonious identification of input variables and lags to forecast an unknown data generating process without domain knowledge poses a key problem in model specification [1, 3]. This becomes particularly important, as complex time series components may include deterministic or stochastic trends, cycles and seasonality, interacting in a linear or nonlinear model with pulses, level shifts, structural breaks and different distributions of noise. Although a number of statistical methods have been developed to support the identification of linear dependencies, their use in nonlinear prediction has not been investigated in detail.

Therefore a structured evaluation of different methodologies to specify the input vector of NN in time series forecasting is required. This paper contributes to the discussion, presenting an analysis of different methodologies of input variable identification through an empirical simulation on the ESTSP forecasting competition time series. This paper is organized as follows. First, we briefly introduce NN in the context of time series forecasting and various methodologies for input variable

identification. Section III presents the experimental design and the results obtained. Finally, we provide conclusions and future work in section IV.

2. Methods

2.1 Forecasting with Multilayer Perceptrons

Forecasting with NNs provides many degrees of freedom in determining the model form and input variables to predict a dependent variable $\hat{y}$. Through specification of the input vector of n lagged realisations of only the dependent variable y a feedforward NN can be configured for time series forecasting as $\hat{y}_{t+1} = f(y_t, y_{t-1}, ..., y_{t-n+1})$, or by including i explanatory variables x_i of metric or nominal scale for causal forecasting, estimating a functional relationship of the form $\hat{y} = f(x_1, x_2, ..., x_z)$. By extending the model form through lagged realisations of the independent variables $x_{i,t-n}$ and dependent variable y_{t-n} more general dynamic regression and autoregressive (AR) transfer function models may be estimated. To further extend the autoregressive model forms of feedforward architectures, recurrent architectures allow incorporation of moving average components (MA) of past model errors in analogy to the ARIMA-Methodology of Box and Jenkins [4]. Due to the large degrees of freedom in modelling NN for forecasting, we present a brief introduction to specifying feedforward NN for time series modelling; a general discussion is given in [5, 6]. Forecasting time series with NN is generally based on modelling a network in analogy to an non-linear autoregressive AR(p) model using a feed-forward Multilayer Perceptron (MLP) [1]. The architecture of a MLP of arbitrary topology is displayed in figure 1.

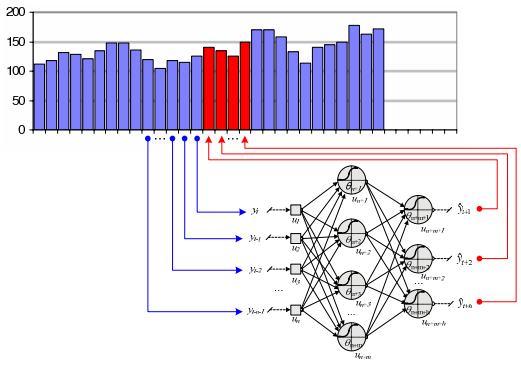

Fig. 1: Autoregressive MLP for time series forecasting.

In time series prediction, at a point in time t, a one-step ahead forecast $\hat{y}_{t+1}$ is computed using $p=n$ observations $y_t, y_{t-1}, \ldots, y_{t-n+1}$ from n preceding points in time t, t-1, t-2, …, t-n+1, with n denoting the number of input units of the ANN. Data is presented to the MLP as a sliding window over the time series observations. The task of the MLP is to model the underlying generator of the data during training, so that a valid forecast is made when the trained ANN network is subsequently presented with a new input vector value [5].

The network paradigm of MLP offers extensive degrees of freedom in modeling for prediction tasks. Structuring the degrees of freedom, each expert must decide upon the selection and sampling of datasets, the degrees of data preprocessing, the static architectural properties, the signal processing within nodes and the learning algorithm in order to achieve the design goal, characterized through the objective function or error function. For a detailed discussion of these issues and the ability of NN to forecast univariate time series, the reader is referred to [1]. The specification of the input vector has been identified as being particularly crucial to achieving valid and reliable results, and will be examined in the next section.

2.2 Input Variable Selection for Multilayer Perceptron predictions

The identification of relevant input variables and variable lags aims at capturing the relevant components of the data generating process in a parsimonious form. In time series modeling, it is closely related with identifying the underlying time series components of trend and seasonality and capturing their deterministic behavior in lags of the dependent variable. A simple visual analysis of the time series components frequently fails to reveal the complex interactions of autoregressive and moving average components, multiple overlying and interacting seasonalities and nonlinear patterns. Several methodologies have been developed for input variables selection of the significant lags in forecasting, originating from linear statistics and engineering. However, currently no uniformly accepted approach exists to identify linear or nonlinear input variables [1].

Seasonality is frequently identified following the Box-Jenkins methodology of linear statistics [4] as a mixture of autoregressive and moving average components. The specification of a parsimonious input vector requires a stepwise analysis of the patterns in the plotted autocorrelation function (ACF) and partial autocorrelation function (PACF) to identify statistically significant autoregressive lags of the dependent variable and of moving average lags of the errors of past predictions. The iterative methodology is frequently employed in identifying significant lags for NN forecasting, following Lachtermacher and Fuller [7]. As in detrending, no consensus exits on whether a time series with identified sesonality should be deseasonalised first to enhance the accuracy of NN predictions [3, 8, 9] or seasonality be incorporated as AR- and MA-components in the NN structure [10-13]. Earlier studies in MLP modeling claim that an analysis of the AR-terms purely from PACF-analysis is sufficient to identify the relevant lags of the time series [14]. However, an AR-analysis can only reveal linear correlations within the time series structure, but not of linear moving average components that require the use of recurrent NN architectures. In addition, ACF and PACF analysis allow no identification of nonlinear interdependencies [1]

In addition, spectral analysis (SA) may provide additional information on the linear autoregressive structure of multiple seasonalities with overlaying periodicities in comparison to an ACF - & PACF-analysis [15], albeit losing information on the potential moving average structure. SA expresses a time series as a number of overlaid sine and the cosine functions of different length or frequency. It identifies the correlation in a periodigram using Fast Fourier Transforms (FFT), which plot the power spectral density versus the frequency of the signal to identify frequencies of

high power as an indication of a strong periodicity. To recode the power spectrum as lags instead of frequencies, we plot the horizontal axis of the periodogram as $n/2$ lags of the time series. This allows a direct association of the power and lags, since the power is expressed in non-continuous terms and directly associated with a specified lag. Significant spikes in the periodigram identify interrelations as input lags for the NN model, which will allow the network to learn and extrapolate the overlaying periodicities. Consequently, SA can be employed in analogy to the ARIMA-methodology to identify periodicities and lags in the time series.

3. Experimental Design

3.1 Exploratory Data Analysis

A single time series of 875 observations, displayed in fig. 1, was provided for the forecasting competition of the 2007 European Time Series Symposium (ESTSP).

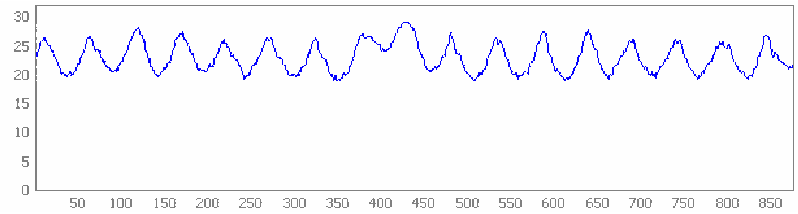

Fig. 2: ESTP2007: Competition time series

The ESTSP competition evaluates the forecasting accuracy on a single time series from a single time origin using the mean squared error (MSE) on the next 15 and the next 50 observations. No domain knowledge was provided to aid the identification of a suitable input vector or network architecture, making the selection of input variables one of the core problems in the competition task. Several different modelling approaches were evaluated, including visual analysis, Autocorrelation analysis and Spectral analysis using FFT.

A visual analysis of the time series reveals a non-trended, seasonal structure of approximately 52 observations with a high signal to noise ratio and a single seasonal outlier that promises easy approximation and extrapolation with a deterministic sine function and exogenous outlier correction. A further visual analysis of the repeating sine pattern displays the repetitive structure in a seasonal diagram, overlying each season containing 52 observations as separate time series displayed in fig. 2. The seasonal diagram confirms a general seasonal structure and two outliers. Although the series seems to obey a 52 observation seasonal length, the peaks of multiple series do not correlate adequately as visualised by the horizontal shift of the series. In contrast to a vertical variation caused by the inherent randomness of the series this indicates an inconsistent or shifting seasonal pattern. The length of 16.8 seasons in the complete series also suggests a seasonality of varying length, rather then an incomplete time series with 9 missing observations. To evaluate this a simple benchmark using a 52 period seasonality in a t-52 lag structure is modelled and evaluated in MLP_{NAIVE} .

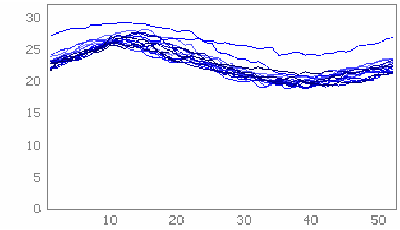

Fig. 3: Seasonal diagram of a 52 observations length

Furthermore, the time series was annotated in a descriptive analysis, to analyse the properties of the seasonal structure of the time series in further detail. Fig. 3 shows the time series plot overlayed with a sine of 52 observations period length.

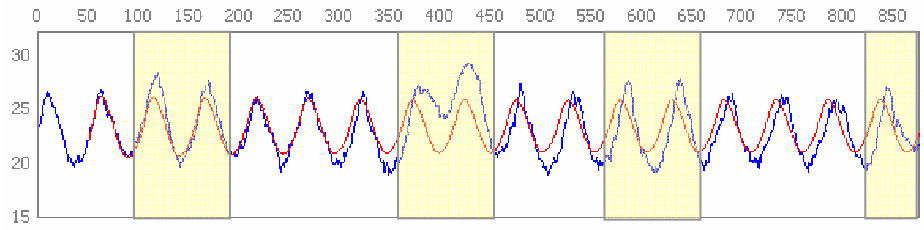

Fig. 4: Time series overlaid by a 52 observations period sine.

Two anomalous seasonal patterns are evident from observation 350 to 450 and in the last season from 825 to 875. Before each anomalous pattern 7 normal seasons are identifiable, with a further pattern of 2 seasonal peaks with increased magnitude every 2 and 3 seasonal patterns apart, indicated as shaded seasons in fig. 3. Although this may suggest a structural pattern in the data generating process, and that an important deviation from the more general model form may be expected for the final ex ante forecasted seasonality, too limited evidence of 1 full cycle is provided. Hence we must consider leaving the anomalies as part of the general model structure or modelling them as outliers using binary dummy variables for NN predictions.

Additionally, a comparison of the time series in fig.3 and a sine function with constant seasonality of 52 observations reveals a varying length of the seasonal patterns. To quantify the pattern of varying seasonal length we estimate the number of observations between each seasonal maxima and minima, applying a 9 period moving average to smooth out randomness. The variation of seasonal lengths in Table 1 shows an average length of 52.06 observations between minima with a standard deviation of 2.59 observations and a significant range of up to 10 observations.

Season	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Length	52	52	51	51	53	51	55	53	50	51	56	49	50	53	58	48

Table 1: Varying seasonal lengths of the time series.

The average length of the seasonality of 52.06 is supported by the visual analysis of the seasonal pattern. However, it biases the identification through ACF and PACF analysis as well as the SA periodigrams, as the temporal interdependencies vary along the time series. Also, the varying seasonal length provides problems as

conventional MLP models assume an AR(p)-process with a deterministic seasonality in an input vector of fixed length. However, the varying seasonality appears to be not entirely stochastic, as a plot of the seasonal lengths in Fig. 4 suggests a regular pattern that may allow exploitation to predict the seasonal length of the forecasted period through the model form or an explanatory variable.

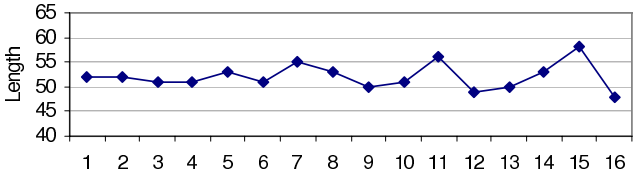

Fig. 5: Length of the time series "sine blocks"

In contrast, the autocorrelation analysis following the Box-Jenkins methodology reveals contradicting information on the more complex structure of the time series. The analysis of the ACF and PACF patterns provided in figure 5a and 5b allow an iterative identification of the significant seasonal or shorter lags using single or seasonal differencing.

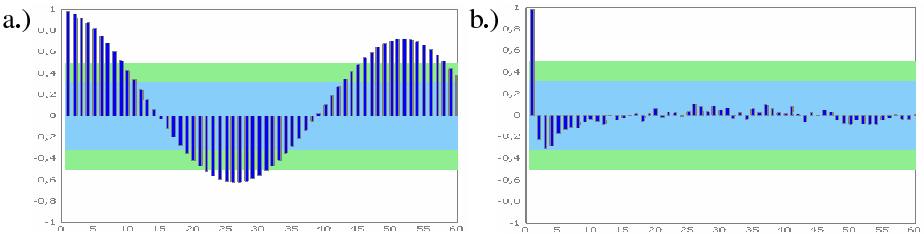

Fig. 6: ACF plot (a.) and PACF plot (b.) of the ESTSP competition time series

The information on the seasonal structure derived from fig. 5 is ambiguous. The ACF plot in fig. 5a reveals a significant seasonal autoregressive process in a decaying, sinusoid pattern of the ACF of a length shorten then 52 periods. In contrast, in the PACF of figure 3b only the first lag is found to be statistically significant at a 0.95 level, and no significant lags are identified around the 26th or 52nd lag. Hence we can not conclude a statistically significant linear seasonal autoregressive process of length 52 from the ACF analysis, despite the series visual appearance. In addition, no moving average process is identified either. An augmented Dickey-Fuller unit root test confirms the stationary form of the time series; hence further differencing provides to no additional information. Consequently, the Box-Jenkins methodology does not allow valid and reliable identification of the model form of this time series, which will later be reflected in the poor performance of the MLP_{ACF} candidate models created using the input vector identified by the ACF-Analysis.

To further analyse the periodicity of the time series a spectral analysis was conducted to reveal information that the autocorrelation analysis may have missed. A variation of a periodogram shows the first 60 lags instead of the frequency along the horizontal axis in fig. 6, as the remaining lags were found to be insignificant.

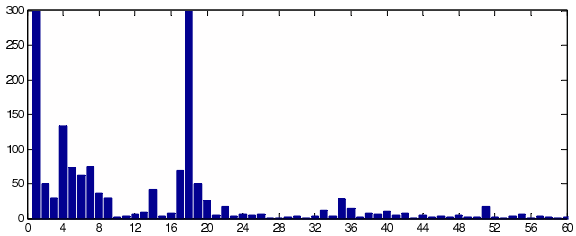

Fig. 7: Periodogram expressed in lags for the first 60 lags.

The periodigram in fig. 6. identifies the 1st and the 18th lag as highly significant, plus a set of additional lags {2, 3, 4, 6, 7, 8, 9, 14, 17, 19, 20, 22, 34, 35, 51} to be of lesser significance. Interestingly, the 52nd lag is again insignificant despite the visual appearance of a 52-observation seasonality. In contrast, lag 51 and preceding lags are found to be significant, which contradicts the analysis in the seasonal diagram in fig. 2 and the ACF and PACF cycles of 26 observations in figure 3a. As different approaches of data exploration lead to different input vectors we consider both lag-groups as candidate models MLP_{FFT} in the later evaluation to build MLP forecasts.

3.2 Artificial Neural Network Models

We create a NN model for each of the three candidate methods of data exploration. First, based upon the visual analysis of the seasonal diagram an input vector containing only the last seasonal lag of the dependent variable y_{t-51} is created, named MLP_{NAIVE} according to the seasonal Naïve forecasting method [16] which serves as a benchmark. In addition, two NN candidate architectures using the input vector identified by the Box-Jenkins methodology of autocorrelation analysis are created, using lags of $\{y_t, y_{t-1}, y_{t-2}\}$ named MLP_{BJ-1} and using $\{y_t, y_{t-1}, y_{t-2}, y_{t-51}\}$ named MLP_{BJ-2} including the plausible ACF information found statistically insignificant in the PACF function. Using the spectral analysis and FFT to determine input lags, three distinct candidate models were created, using additional information on the time variation of the seasonality and the outlier seasons. First, a basic MLP_{FFT} was created using an input vector of the significant lags as identified by the SA $\{y_t, y_{t-1}, \ldots, y_{t-8}, y_{t-13}, y_{t-16}, \ldots, y_{t-19}, y_{t-21}, y_{t-33}, y_{t-34}, y_{t-50}\}$. In addition, a MLP using only the highly significant lags of $\{y_t, y_{t-17}\}$ is evaluated but discarded due to significantly inferior results.

In order explicitly model the varying length of he seasonal cycles in the time series an explanatory variable is created to encode the seasonal length. Using the number of observations between consecutive minima in table 1 the relative position of each observation in each season was calculated. We divided an arbitraty number of 100 by the number of observations per season, creating a time series of {2, 4, 6, …, 100} for a season with 50 observations, {1.851, 3.703, 5.555, …, 100} for a season with 54 observations etc. Essentially, this created a temporal mapping that translated the relative position of each observation in a season of varying length onto a stationary level. A MLP_{TEMP} using the temporal encoding as an explanatory time series x_t was created, using only the explanatory variable $\{x_{t+1}\}$ in $t+1$ as an input. In addition, a topology using the lags identified from the FFT was created utilising the lags only for the dependent time series $\{y_t, y_{t-1}, \ldots, y_{t-8}, y_{t-13}, y_{t-16}, \ldots, y_{t-19}, y_{t-21}, y_{t-33}, y_{t-34}, y_{t-50}\}$ and $\{x_{t+1}\}$ for $MLP_{TEMP\text{-}FFT\text{-}1}$, using the FFT lags only on the explanatory

time series of the temporal encoding $\{x_t, x_{t-1}, \ldots, x_{t-8}, x_{t-13}, x_{t-16}, \ldots, x_{t-19}, x_{t-21}, x_{t-33}, x_{t-34}, x_{t-50}\}$ for $MLP_{TEMP\text{-}FFT\text{-}2}$, and using the FFT lags on both the time series y_t and the temporal encoding x_t for $MLP_{TEMP\text{-}FFT\text{-}3}$.

In order to eliminate the impact of the two abnormal seasonal profiles in the mid section of the time series a binary dummy variable $z_t \in \{0, 1\}$ was created with the value '1' for the two abnormal seasons and '0' otherwise. An input vector using only contemporaneous realizations of the explanatory variables for temporal encoding x_t and the time series of the binary dummies z_t was created $\{x_{t+1}, z_{t+1}\}$ for $MLP_{BIN\text{-}TEMP}$. In addition, corresponding topologies using the identified lags from FFT analysis were created for only the dependent variable y_t as $MLP_{BIN\text{-}TEMP\text{-}FFT\text{-}1}$, using the FFT lags for both time series of the dependent variable y_t and time mapping x_t as $MLP_{BIN\text{-}TEMP\text{-}FFT\text{-}2}$ and a topology using the FFT lags only for the explanatory series for time mapping as $MLP_{BIN\text{-}TEMP\text{-}FFT\text{-}3}$. Finally, we created a topology using the FFT lags on all three time series y_t, x_t and z_t as $MLP_{BIN\text{-}TEMP\text{-}FFT\text{-}4}$.

For the comparative analysis of alternative input vectors prior to the final predictions the time series was sequentially split into 60% observations for training, 20% for validation and 20% for out of sample testing. All data was linearly scaled into the interval of -0.6 to 0.6 to avoid saturation effects of the activation functions. As no indication for a MA-process that would require recurrent topologies could be determined from the ACF & PACF data analysis, we limited our evaluation to feedforward architectures of MLP. All MLPs architecture contained a single output node for iterative one-sep ahead forecasts up to 50 steps into the future, $\hat{y}_{t+1}, \hat{y}_{t+2}, \ldots, \hat{y}_{t+50}$. For each MLP candidate, we evaluate topologies with 1 … 20 hidden nodes in steps of 4 for a single and two hidden layers. Each network was initialised 20 times with randomised starting weights to account for local minima. It was then trained for 1000 epochs on minimising the final evaluation criteria MSE using the backpropagation algorithm with an initial learning rate of η=0.5 that was decreased by 1% every epoch. Training was terminated using early stopping if the MLP did not decrease the MSE over 0.1% in 100 epochs. A composite error of 30% training MSE and 70% validation MSE was used to avoid overfitting effects on the validation set in early stopping. We select the network topology and initialisation with the lowest composite early stopping error and evaluate its accuracy. All MLP models were calculated using the software BISlab Intelligent Forecaster (IF).

4. Experimental Results

The experimental results provided in table 2 give an overview of the criteria used to specify the input vector for the dependent variable y_t, the explanatory variable mapping temporal seasonal lengths x_t and the binary variable for outlier mapping z_t.

Although the simple approach of a seasonal naïve model MLP_{NAIVE} demonstrated adequate accuracy on validation and test data, a visual inspection of the predictions showed unsatisfactory results of extrapolating only a simple sine pattern, without replication of the outliers or the shifting seasonality. However, in accordance with established practice in forecasting a naïve approach may serve as a parsimonious benchmark to compare potential improvements of more complex model forms.

	y_t Predictor Variable	x_t Time Mapping	z_t Outlier Coding	MSE Train	MSE Valid	MSE Train &Valid	MSE Test
MLP_{NAIVE}	t-51	-	-	3.95	0.86	1.79	0.75
MLP_{BJ-1}	BJ-1	-	-	6.24	6.20	6.21	4.75
MLP_{BJ-2}	BJ-2	-	-	5.12	4.39	4.61	2.92
MLP_{FFT}	FFT	-	-	1.70	0.81	1.08	2.12
MLP_{TEMP}	-	t+1	-	2.23	0.46	0.99	0.55
$MLP_{TEMP-FFT-1}$	FFT	t+1	-	1.38	0.68	0.89	0.86
$MLP_{TEMP-FFT-2}$	-	FFT	-	1.53	0.64	0.97	0.72
$MLP_{TEMP-FFT-3}$	FFT	FFT	-	1.60	0.70	0.91	3.92
$MLP_{BIN-TEMP}$	-	t+1	t+1	1.32	0.43	0.70	0.52
$MLP_{BIN-TEMP-FFT-1}$	FFT	t+1	t+1	0.60	0.60	0.60	0.33
$MLP_{BIN-TEMP-FFT-2}$	FFT	FFT	t+1	0.44	0.55	0.51	0.68
$MLP_{BIN-TEMP-FFT-3}$	-	FFT	t+1	1.21	0.45	0.68	0.61
$MLP_{BIN-TEMP-FFT-4}$	FFT	FFT	FFT	0.41	0.70	0.55	0.59

Table 2: MLP candidate inputs and MSE on training, validation and test set

Both MLP_{BJ-1} and MLP_{BJ-2} using the Box-Jenkins methodology for input vector specification failed to approximate the shifting seasonality or the anomalies in the training set. As a consequence, the provided only a smooth sine curve with dampening magnitude on the validation and test set, leading to higher MSE then the MLP_{NAIVE} benchmark on all data subsets. Due to the nature of the varying seasonality, the specification of alternative lag structures did not increase accuracy either.

Similarly the MLP_{FFT} using SA and FFT to identify potential multiple overlying seasonalities failed to generalize on the test set, although providing significantly better results in approximating the pattern in sample. as indicated by the lower in sample errors on training and validation set. Again, the MLP were unable to capture the anomalous observations and the shifting seasonal length across different initializations and topologies, justifying a different modeling approach in providing additional information on seasonal length through explanatory variables.

The MLP_{TEMP} topologies using the temporal encoding x_t as a causal variable in t+1 reduced MSE in sample and out of sample, supporting the importance of external coding of shifting seasonal lengths. The forecasts showed a repeating sine-pattern of varying seasonality, closely resembling the observed time series frequencies. However, the MLP_{TEMP} failed to capture some subtle repetitive patterns that previous models using FFT lags had been able to approximate. In contrast, the $MLP_{TEMP-FFT-1}$ utilizing the lags identified from SA on the time series of the dependent variables $\{y_t, y_{t-1}, \ldots, y_{t-8}, y_{t-13}, y_{t-16}, \ldots, y_{t-19}, y_{t-21}, y_{t-33}, y_{t-34}, y_{t-50}\}$ plus a temporal coding showed little error improvement in comparison to MLP_{TEMP}. However, providing the FFT lags only on the temporal variable $\{x_t, x_{t-1}, \ldots, x_{t-8}, x_{t-13}, x_{t-16}, \ldots, x_{t-19}, x_{t-21}, x_{t-33}, x_{t-34}, x_{t-50}\}$ for $MLP_{TEMP-FFT-2}$ allowed a closer approximation of different periodicities and a significant increase in accuracy on the hold out data of the test set.

Despite reduced errors the seasonal anomaly observed in the time series could not be explained and negatively affected the accuracy of the approximation in sample. As only a single anomaly could be observed and no MLP model had shown the capability of approximating it as part of the data generating process, the lack of further evidence suggested an exclusion of these outliers from model building using a binary variable in addition to the previous models of temporal encoding and FFT lags. The the binary outlier variable in $MLP_{BIN-TEMP}$ enhanced the in sample approximation

and reduced the training MSE significantly. In addition, it further reduced the errors on the validation and test set in comparison to the previous topology of MLP_{TEMP}. A use of the dynamic FFT lags on the three variables *y*, *x* and *z* resulted in the selection of $MLP_{BIN\text{-}TEMP\text{-}FFT\text{-}2}$ with the lowest composite error of in sample approximation and out of sample generalization for the final forecasts. Fig. 7 illustrates the models iterative $t+1, \ldots t+50$ step ahead prediction of multiple overlaying 50 period ahead forecasts originating from each point of the time series. The graph shows that the MLP has adequately learned the pattern on training and validation set, including the abnormal seasonality coded as outliers, except the last seasonal pattern also possibly containing an outlier.

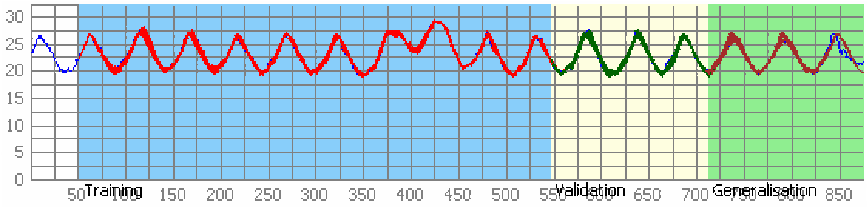

Fig. 8: 50 periods forecasts originating from each point of the time series

The selected $MLP_{BIN\text{-}TEMP\text{-}FFT\text{-}2}$ utilises the 16 lags identified by the FFT for the dependent variable y_t, the explanatory variable of temporal coding x_t and a single explanatory dummy variable to encode the outlier z_{t+1}, constructing an input vector of 37 variables. The MLP uses two hidden layers of 20 nodes each with a logistic activation function and a single output node with the identify function. The model is used to compute the final forecasts 50 steps ahead, as shown in fig. 8.

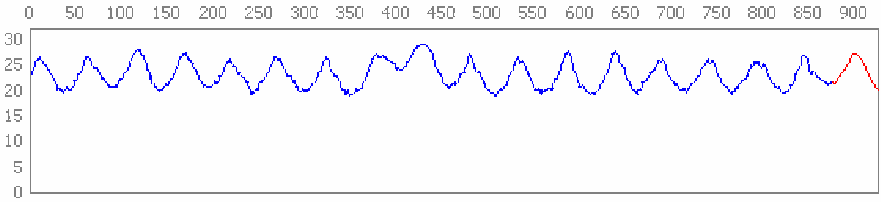

Fig. 9: The time series is plotted with the forecasts

The final ex ante forecasts required the prediction of the temporal explanatory variable beyond the provided dataset. A decision upon the position of the minimum of the last observable season and the expected length of the next seasonal cycle outside the provided data was based upon the regularity in seasonal cycle length observed in fig. 4 and the ESTSP competition objectives. Hence the next seasonal cycle was expected to be 50 observations for the final ex ante forecasts.

5. Conclusions

We evaluate a number of conventional methodologies of visual inspection, autocorrelation analysis and spectral analysis to specify significant input variables for NN prediction on the ESTSP competition time series. Due to the particular nature of

the series, containing a seasonal pattern with varying length and anomalous observations, the conventional approaches fail to specify adequate input variable lags.

To compensate for this we propose a dynamic causal modelling approach, coding the shifting seasonal cycle length and the outliers in explanatory variables, utilising the same temporal lag structure as identified in the original time series using spectral analysis. Although ACF & PACF analysis as well as SA fail to identify the input lags, they are frequently applied in NN modelling where they have a proven track record in identifying seasonal patterns of constant cycle length. Hence the results provided here should not be generalised beyond the single time series. In comparison, SA based upon FFT periodigrams demonstrate a better performance in extracting more information regarding periodic effects from this time series. However, this may again prove misleading for moving average processes which require identification in ACF-plots and subsequent modelling with recurrent NN.

Fur future research, a systematic evaluation of methodologies for input lag identification is required, extending the analysis to multiple time series, multiple time origins to increase generalisation and to unbiased error metrics, avoiding over penalisation of high deviations and outliers from squared error measures.

References

[1] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting*, vol. 14, pp. 35-62, 1998.

[2] S. Makridakis and M. Hibon, "The M3-Competition: results, conclusions and implications," *International Journal Of Forecasting*, vol. 16, pp. 451-476, 2000.

[3] T. Hill, M. O'Connor, and W. Remus, "Neural network models for time series forecasts," *Management Science*, vol. 42, pp. 1082-1092, 1996.

[4] G. E. P. Box and G. M. Jenkins, *Time series analysis: forecasting and control*. San Francisco: Holden-Day, 1970.

[5] C. M. Bishop, *Neural networks for pattern recognition*. Oxford: Oxford University Press, 1995.

[6] S. S. Haykin, *Neural networks: a comprehensive foundation*, 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 1999.

[7] G. Lachtermacher and J. D. Fuller, "Backpropagation in time-series forecasting," *Journal of Forecasting*, vol. 14, pp. 381, 1995.

[8] M. Nelson, T. Hill, W. Remus, and M. O'Connor, "Time series forecasting using neural networks: Should the data be deseasonalized first?" *Journal of Forecasting*, vol. 18, pp. 359-367, 1999.

[9] G. P. Zhang and M. Qi, "Neural network forecasting for seasonal and trend time series," *European Journal Of Operational Research*, vol. 160, pp. 501-514, 2005.

[10] L. Zhou, F. Collopy, and M. Kennedy, "The Problem of Neural Networks in Business Forecasting - An Attempt to Reproduc th Hill, O'Connor and Remus Study," CWR, Cleveland 2003 2003.

[11] S. F. Crone, J. Guajardo, and R. Weber, "The impact of Data Preprocessing on Support Vector Regression and Artificial Neural Networks in Time Series Forecasting," presented at World Congress in Computational Intelligence, WCCI'06, Vancouver, Canada, 2006.

[12] S. F. Crone, J. Guajardo, and R. Weber, "A study on the ability of Support Vector Regression and Neural Networks to Forecast Basic Time Series Patterns," presented at IFIP World Congress in Computation, WCC'06, Santiago, Chile, 2006.

[13] S. F. Crone, S. Lessmann, and S. Pietsch, "An empirical Evaluation of Support Vector Regression versus Artificial Neural Networks to Forecast basic Time Series Patterns," presented at World Congress in Computational Intelligence, WCCI'06, Vancouver, Canada, 2006.

[14] Z. Y. Tang and P. A. Fishwick, "Feed-forward Neural Nets as Models for Time Series Forecasting," *ORSA Journal on Computing*, vol. 5, pp. 374-386, 1993.

[15] S. M. Kay and S. L. Marple, "Spectrum Analysis - A Modern Perspective," *Proceedings Of The Ieee*, vol. 69, pp. 1380-1419, 1981.

[16] S. G. Makridakis, S. C. Wheelwright, and R. J. Hyndman, *Forecasting: methods and applications*. New York: Wiley, 1998.

Bifurcation Forecasting Using Time Series Statistical Analysis

M.-G. M. Zulpukarov, G.G. Malinetskii, A.V. Podlazov

Keldysh Institute for Applied Mathematics Russia Academy of Science – Department of mathematical modelling of nonlinear processes and synergetics
Miusskaya sq., 4, Moscow, 125047, www.keldysh.ru – Russia

Abstract. A nonlinear dynamical system behavior near the supercritical pitchfork bifurcation point is considered. The time series statistical characteristics dependency on the bifurcation parameter is given. The bifurcation forecasting using the time rows statistical characteristics analysis is discussed.

Introduction

Nowadays, the nonlinear mathematical models are widely used for researching the complicated systems behaviour in the various areas of exploration, including the ones related to the sciences still not considered as exact sciences: medicine (cardiology, psychology), sociology, economics, history, and so on. A bifurcation of a nonlinear system in a model like that can have an important practical meaning. For instance, the following fact is known: if a system is in a stable state, then a considerable effort is needed to transfer this system to another stable state. Practically, this means the following: if a controlled object is in a stable state, and this state is not appropriate for us, then in many cases we are not able tho change the state of the object.

A fundamentally different case is observed, when the system goes through a bifurcation point. In this case there is a possibility to choose the way of further evolution, by making comparably small effort. Here, the two classes of the practical problems are arised. First, a retrospective analysis of the system behaviour: detecting the critical moments in the past, and researching the possible alternative ways of evolution (the analysis problem). Second, forecasting the bifurcations, and applying the controlling efforts in the appropriate direction (the synthesis problem).

The following question emerges naturally: which universal and stable signs accompany a bifurcation, and do these signs become apparent before? Now it is known, that these questions can be answered studying the behaviour of the system undergoing a weak noise. Let the studied system be modelled by a system of nonlinear ordinary differential equations (or nonlinear maps) disturbed by a weak noise. Nowadays, it is shown for some different cases, that in the neighborhood of a bifurcation point, noise in the system tends do grow [1,2,3]. Probably, this phenomenon is universal.

Thus, the problem can be stated the following way. First, according to the specific of the studied system, determine, what can be considered as noise, and which way it can be measured (for instance, in the case of studying a society, the minor crimes, or the dynamics of the printed periodicals replication, or the career dynamics in the political structures, can be considered as a weak noise). Then, retrospectively research the system behaviour, and observe the anomalous growth of the noise, to

check, whether the right choice of the noise was made, and to receive some additional information about the system and its parameters. Next, the information received can be used for prognosis. The prognosis accuracy will depend on the amount of the information.

Time series statistical characteristics near the bifurcation point

The present work discusses the noise rise phenomenon in the case of the bifurcation in the simple ordinary differential equation

$$\dot{x} = x(\lambda - x^2), \tag{1}$$

where λ is a parameter. Increasing the parameter, at the critical point $\lambda = 0$ the supercritical pitchfork bifurcation occures: the $x = 0$ stable state loses its stability, and the two new stable states appear, $x = \pm\sqrt{\lambda}$.

Let the system (1) undergo a *brownian noise*: at the moments t_i some random number ξ is added to the x variable. Also, let the noise be uniformly distributed and periodic:

$$f_\xi = \begin{cases} 1/A, & |\xi| \le A/2 \\ 0, & |\xi| > A/2 \end{cases}$$

$$\sigma_\xi = \frac{A}{2\sqrt{3}}$$

$$t_{i+1} - t_i = T_\xi, \quad i = 0,\ 1,\ 2,\ \ldots$$

Thus, $x(t)$ will be a kind of random process, which will form some input time series $x(t_j)$ (let the discretization be periodic with the period of T_ξ). To derive its statistical characteristics, it is useful to replace a single system (1) by a set of independent systems each undergoing independent noise. This way, the problem can be reduced to considering the diffusion in a potential well. So, the main integral characteristic of the noise is the *coefficient of diffusion*

$$\kappa \equiv \int_0^{+\infty} K(\xi)\xi^2 d\xi,$$

where $K(\xi)$ is the space-time noise distribution:

$$K(\xi) = \frac{1}{T_\xi} f_\xi.$$

By equating the brownian flow caused by the concentration gradient with the flow caused by the potential gradient and taking into account the ergodicity assumption, one can get the expression for $x(t_j)$ distribution

$$f_x = \frac{\exp\left(\frac{2\lambda x^2 - x^4}{4\kappa}\right)}{\int_{-\infty}^{+\infty} \exp\left(\frac{2\lambda x^2 - x^4}{4\kappa}\right) dx}. \quad (2)$$

Finally, $x(t_j)$ variance

$$\sigma_x^2 = \int_{-\infty}^{+\infty} x^2 f_x dx, \quad (3)$$

can be evaluated numerically (fig. 1).

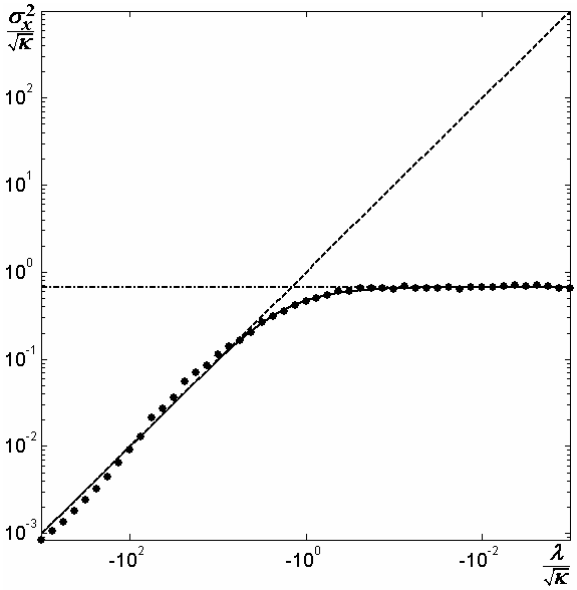

Fig. 1: $x(t_j)$ time series variance growth in the neighbourhood of the bifurcation point. Dots mark the experimental data, dash and dash-dot lines show the asymptotes. Solid line denotes the numerical evaluation of (3).

The numerical simulation results showed, that the $x(t)$ signal distribution changes in the neighborhood of the bifurcation point according to the equation (2). It is worth mentioning, that the qualitative character of the system behavior is similar for the case of another kind of noise (for example, the normally distributed one) or different discretization frequency.

Conclusion

Thus, when it is known, that the supercritical pitchfork bifurcation can emerge in the system, then its moment can be forecast using formulas (2) and (3) applied to the time series statistical characteristics.

References

[1] Yu.A. Kravtsov, S.G. Bilchinskaya, O.Ya. Butkovskii, I.A. Rychka, E.D. Surovyatkina. Prebifurcational Noise Rise in Nonlinear Systems. *Journal of Experimental and Theoretical Physics*, Vol. 93, No. 6, 2001, pp. 1323-1329.

[2] E.D. Surovyatkina. Rise and saturation of the correlation time near bifurcation threshold. *Physics Letters A*, 329 (2004), pp. 169-172.

[3] A. Juel, A.G. Darbyshire, T. Mullin. The effect of noise on pitchfork and Hopf bifurcations. *Proc. R. Soc. Lond. A*, 453 (1997), pp. 2627-2647.

Modeling the Usage Characteristics of Content Services

Elizabeth A. Anglo

Ateneo de Manila University - Dept of Information Systems and Computer Science
Loyola Heights, Quezon City - Philippines

Abstract. The goal of data analysis in a business organization is simple: facilitate effective decision-making in terms of resource allocation and marketing strategies. In this paper, the usage characteristics of the customers of a content provider are analyzed and modeled. Such analysis and model may be used to forecast the network requirements and strategically schedule marketing campaigns. Initial results suggest that demand for content services exhibit time-of-day and day-of-month sensitivities, and seasonality.

1 Introduction

The Philippines has earned the distinction of being the text capital of the world. In a country where there are 10 times more mobile lines than fixed telephone lines and an estimated 250 million text messages exchanged daily [1], this distinction is not at all surprising. To the Filipino, sending text messages is a very economical and convenient form of communication.

The Filipino's fascination with the mobile device did not escape the attention of enterprising businessmen. It was an opportunity to earn big revenues while providing new forms of entertainment, information-on-demand and social interaction. A new business form called content provision was introduced to the market. Content providers or CPs create applications called value-added services (VAS) which are accessible to cellular phone users via subscription. The value-added services easily became a craze making the CP business a competitive arena. As of the moment, there are over a hundred CPs operating locally with the top ones having about 80,000 subscribers.

Being in such a competitive market, content providers need to understand the traffic patterns of usage of their subscribers. Knowledge of traffic patterns is useful for network design and planning. For example, information about the bandwidth usage patterns for the different services of telecommunications companies helped determine the future network architectures and resources based on current demands [2, 3, 4].

This paper aims to model the behavior of usage for the services of a content provider. Using time-series data, it will examine this behavior to see whether it is subject to trend, seasonality, and sensitivities to the time-of-day, the day-of-week and the day-of-month. Finally, a model will be constructed to fit the time-series data. This model can later be used to forecast the future usage of the services.

1.1 Statement of the Problem

This paper aims to model the traffic patterns for content services. In particular, the following questions are to be answered:

1. What trends are exhibited by the usage? By examining the trends exhibited, the business manager can predict whether the demand for value-added services follows an upward or downward trend.
2. Do the time series exhibit time-of-day/day-of-week/day-of-month sensitivity in usage that is similar to sensitivities found in other related industries [3]? Knowing these sensitivities is helpful for three reasons:
    - By knowing the time-of-day sensitivities, the service provider can dimension the network load to accommodate peak times since quality of service declines when the network is overloaded;
    - By knowing the day-of-week sensitivities, maintenance operations can be scheduled so that they do not hamper the overall quality/delivery of service;
    - By knowing the day-of-month sensitivity, marketers can schedule their product promotions appropriately.
3. Are there any seasonal patterns in the usage of the services offered by the content provider similar to those found in studies for wireless communication services? Discovery of seasonality will again help marketers in scheduling promotions efficiently.

1.2 Definition of Terms

Forecasting is defined as the prediction of future events by examining the past. For example, stock analysts use various forecasting methods to determine future stock price movements and earnings. Economists too use forecasting techniques in order to determine future economic trends.

There are two general types of forecasting methods: qualitative and quantitative. Qualitative methods make use of human experts' opinions to predict the future values of the object being observed. These methods are limited by the absence of accurate measurement as well as the experts' bias. On the other hand, quantitative methods use statistical analysis of past data at various points in time and thus reliable measures of accuracy exist. This paper uses only quantitative methods to come up with more objective forecasting models.

Quantitative methods are further classified into time-series models and causal models. Time-series models are based on the analysis of a chronological sequence of observations on a particular variable while causal models assume the variable to be forecasted is explained by the behavior of one or more variables.

The operations of a content provider naturally lead to an accumulation of time series data. When a subscriber sends a text message to the content provider to pull the current content of the service he is subscribed to, this is recorded as an incoming text message record. However, only the aggregated count per hour has been used. Aggregation has been used for two reasons: first, to reduce the size of the data and second, to examine attributes of data that are often more stable than that of individual records [5].

There are two aspects in the study of time series – analysis and modeling. In analysis, the objective is to identify the properties of the series. Particular properties focused on are trends and seasonality. Trend is defined as the persistent upward or downward movement while seasonality refers to a pattern of change in the data that completes itself within a calendar year and then is repeated on a yearly basis. Figures 1 and 2 show time-series data with trend and seasonality, respectively.

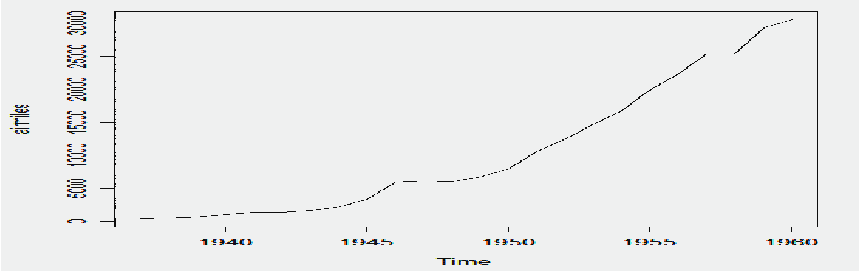

Figure 1. Time-Series with Trend

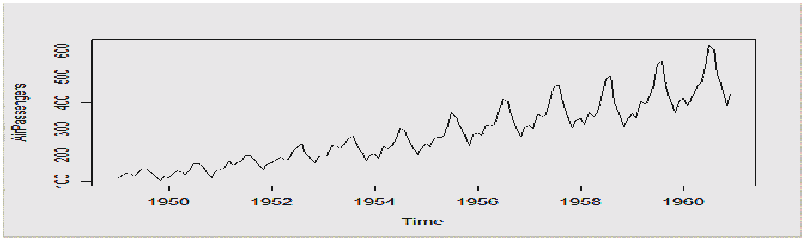

Figure 2. Time-Series with Seasonality

After these properties have been identified, a model will be constructed to represent the behavior of the time-series data. This model may be used to forecast future values.

In finding the best model for a time-series, different measures of accuracy may be employed: mean absolute error (MAE), mean absolute percentage error (MAPE), mean square error (MSE), and root mean square error (RMSE). As a general rule, the smaller the values of these measures, the more accurate is the fit of the model. Another statistical measure of forecast accuracy is the Theill's U which compares the accuracy of a forecast to that of a naïve model. A Theill's U greater than 1 indicates that the forecast is worse than the naïve model; a value less than 1 means the model is better. The closer the Theill's U is to 0, the better the model. In practice, values of 0.55 or less are very good [6].

The problems stated above are time-series problems: the objectives are to look at the characteristics of the time-series data: trend, temporal sensitivity, and seasonality.

2 Review of Related Literature

There are two aspects in the study of time series – analysis and modeling [7]. In analyzing the time series, the objective is to identify the properties of the series; on

the other hand, in modeling, the objective is to be able to use the model to enable forecasting of future values.

Time series modeling consists of the following activities:

1. Plotting the series to examine the main features of the graph. Observation should be focused on the presence of a trend, a seasonal component, and apparent change in behavior, e.g., temporal sensitivity.
2. Usage of decomposition techniques to identify the components of the time-series data: trend, seasonal variations, cycle, and irregular or error fluctuations [6, 8].
3. Once these components are identified, forecasting methods are applied to each component.
4. The combined forecast components provide the time series forecast.

In finding the model for the time-series data or any of its components, several simple forecasting techniques have been used in past researches. One technique, called naïve forecasting, uses the last observed value to predict the future value. The use of moving average model has been shown to be efficient [9]. This model makes prediction based on the mean computed from the most recent data. For example, to construct a three-period moving average, one would use the current observation and the two previous time period observations. A variation of moving average is the centered moving average (CMA). A CMA model centers its average on the current period using both the previous time period observations and the forward time period observations. Another simple forecasting method is the exponential smoothing which is based on the exponentially weighted average of all past observations [10].

3 Hypotheses

So far, no in-depth research has been done on the usage patterns of consumers availing of value-added services in the local setting. Knowledge of this may be useful to content providers in terms of resource allocation and marketing strategies.

This paper attempts to test the following hypotheses:

1. The usage (as exhibited by the number of requests for a VAS) exhibits an upward trend. The lower costs of handsets enable more people to own mobile devices. Hence the potential customer base of content providers is increasing, increasing the demand for value-added services as well.
2. The usage of value-added services exhibits time-of-day and day-of-month sensitivities. Pemmaraju [3] discovered uniform usage pattern on a daily basis among Internet users but varying usage on a per-hour basis. The CP manager observed similar patterns in the demand for value-added services:
    a. The usage attains its peak early in the morning and toward lunch, while the usage for the rest of the day is very low.
    b. The usage on a monthly basis attains its peak during the first and third weeks of the months. It may be speculated that this is due to the fact these periods coincide with the payroll periods of most employees in the country, during which time the people have money to spare in availing of the services.
    c. The CP manager has not expressed any observation of variation in the demand on a weekly basis. However, this paper would like to test

whether such day-of-week sensitivity similar to that found in stock market studies is also present in the demand for content services.

3. The usage of value-added service exhibits seasonality. Kohandani et al [11] observed seasonality in the wireless traffic data. In the case of the content provider, the usage is high during the periods that school is out since young people (who are more likely to be interested to subscribe) will have more time to use the services.

4 Methodology

Data analysis and modeling in time series is accomplished in several stages. Data acquisition for this paper has been done by queries to the CP's database.

For the time series analysis, the counts of the transactions have been aggregated according to the following relevant time dimensions: per hour and per day. Tables 1 and 2 show the data structures returned by the database queries.

Month of Text	Day of Text	Time of Text	Number of Text Messages
June	4	1	5
June	4	2	7
June	4	3	8

Table 1. Structure of Hourly Aggregates Query

Month of Text	Day of Text	Time of Text
June	4	500
June	5	720
June	6	238

Table 2. Structure of Daily Aggregates Query

For example, in Table 1, the first line means that on June 14 at 1 AM, there was a total of five text messages received from the subscribers. On the other hand, in Table 2, the first line means that on June 4, there was a total of 500 text messages received from the subscribers.

Then, a variable transformation (i.e., averaging) has been performed to compress the range of values [5]. The following transaction averages were computed: hourly, daily, monthly, and day-of-week (where 1 is Sunday, 2 is Monday, etc.). These time series data were then plotted on separate graphs, in preparation for analysis and eventual modeling.

A number of possible models will be examined and evaluated for fitness to the time series data as well as in modeling its components. The modeling techniques (naïve forecasting, simple moving average, centered moving average, and exponential smoothing) listed in the literature review will be considered in the analysis. These techniques were chosen because of their strengths as proven in previous researches.

Then the decomposition method will be used to find the trend, seasonal, cyclical, and error components of the time series data in an attempt to find the most suitable model for the data at hand. Since the CP business is a relatively young industry,

cycles in the time series cannot yet be established. Hence the general form of decomposition technique used is expressed as

$$Y_t = f(T_t, S_t, e_t)$$

where Y_t is the time series (or observation) at time t
T_t is the trend component at time t
S_t is the seasonal component at time t
e_t is the error (or residual) at time t

The most common forms of decomposition are the additive decomposition

$$Y_t = T_t + S_t + e_t$$

and the multiplicative decomposition

$$Y_t = T_t \times S_t \times e_t .$$

5 Results and Discussion

The time series data are plotted in Figures 3, 4, 5, and 6. Error bars (using the standard error) were added to graphically show the possible error amounts relative to each data point.

Figure 3 shows that the number of transactions attains its peak at mid-morning, retaining the volume until around 12 PM then slowly tapering off the rest of the day. The graph reveals that from one hour to another in the early morning, the number of transactions is almost uniform, and the same behavior can be observed late in the day, starting at 6 PM. The standard errors show that the neighboring points (e.g., from time 1AM all the way to time 8AM) are not significantly different from each other, with the behavior significantly different starting at 9 AM. This suggests that the demand for content services exhibits time-of-day sensitivity, confirming part (a) of the second hypothesis.

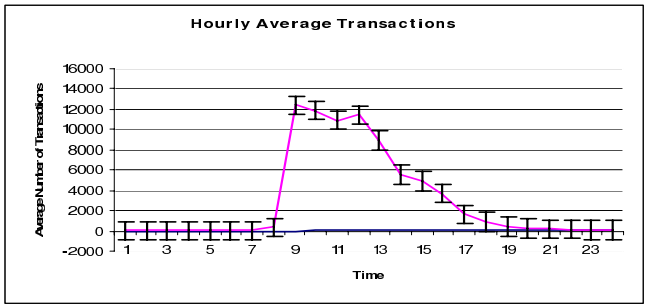

Figure 3. Graph of Hourly Average Transactions.

Figure 4 shows an almost uniform average number of transactions from day to day, except for the 3rd, 4th, 22nd and 30th days of the month. The standard errors show that the neighboring points differ very slightly from each other for most of the days of the month, with the behavior during the aforementioned exceptions changing significantly. This suggests that the demand for content services exhibits day-of-the-

month sensitivity, confirming part (b) of the second hypothesis. However, the sensitivity does not exactly agree with what the CP manager has speculated:

1. Instead of an increased demand during the first week (right after payroll), there was even a decrease!
2. No unusual behavior can be observed during the days of the third week (again, right after the payroll).
3. Instead the change in behavior (a decrease) can be seen during the fourth week. This is probably due to the fact that during the fourth week, people have less money to spare in availing of the services.
4. Finally, an increased demand is observed on the exact date when wages are paid.

For the first two observations, a closer examination of the dynamics of demand will have to be performed to arrive at a more definite conclusion.

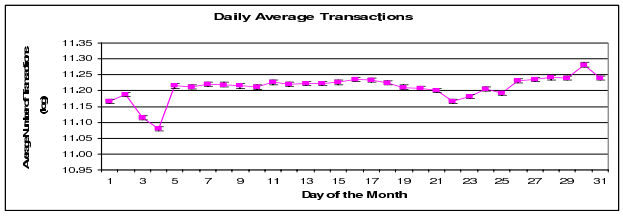

Figure 4. Graph of Daily Average Transactions.

The graph in Figure 5 somehow tells a different story. While the time-of-day and day-of-month sensitivities hypothesized were confirmed by Figures 3 and 4, Figure 5 does not seem to show that there is anything unusual happening on certain days of the week. While the standard errors indicate that the values from one point to another are not always within the range of the possible error in the averages (e.g., days 3, 5, and 7 are significantly higher than their neighboring points), all averages still fall within a small range of values (i.e., from 3,000 to 3,500 transactions). This suggests that there is no day-of-the-week sensitivity, in contrast to that observed for stock markets.

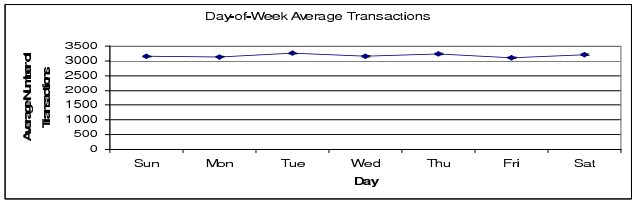

Figure 5. Graph of Day-of-Week Average Transactions.

The graph in Figure 6 shows an increasing demand from the start of the observation period (May 2005) up to March 2006. However, there was a significant drop in the summer months, while the demand picked up again in June up to the end of the observation period (August 2006). The standard errors show that the drops during the summer months, especially in May, were significant. This could suggest that there is seasonality in the demand for content services, i.e., that demand changes significantly during a certain period confirming the seasonality statement in the third hypothesis.

However, again, the direction of the demand is opposite to what has been hypothesized. Could it be that the fact that school is out decreases demand because during this time, the young subscribers do not get allowance? Also, a longer observation period may have to be used to get a firmer basis for the seasonality hypothesis.

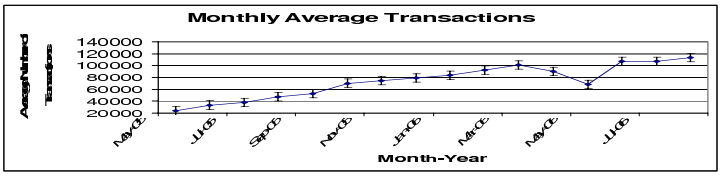

Figure 6. Graph of Monthly Average Transactions.

As has been noted, the behavior from the start of the observation up to the month of March 2006 exhibited an upward linear trend. A sudden change of direction happened during the summer months, after which an increase in demand was again exhibited. To confirm whether the upward linear trend can be concluded, trend analysis using least squares method was employed. Using the R Statistical package, the following results were obtained:

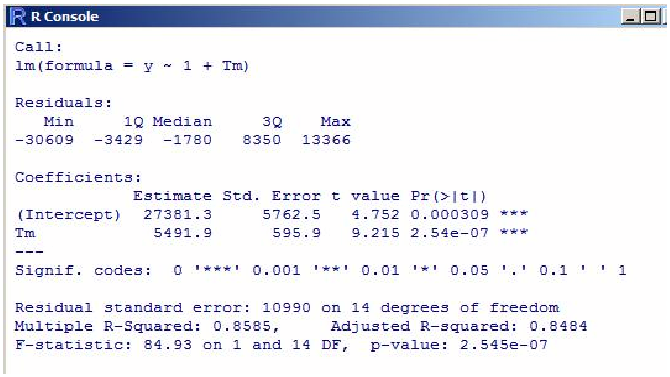

Figure 7. Results for linear trend analysis.

The results in Figure 7 show that there is a statistically significant dependence (at the 1% level, $t_{\alpha,14} = 2.624$) of usage demand with respect to time. The R^2 value also indicates that 86% of the variation in the values of the monthly averages can be explained by the independent variable. Hence there is an upward linear trend confirming the first hypothesis.

A comparison of the different modeling techniques to predict the monthly usage was performed. Monthly usage was the appropriate time period to consider since the dynamics of the CP business changes on a monthly basis. Table 3 shows the results of such comparison:

Measure of Accuracy	Forecast Techniques			
	Naïve	**3-period Simple Moving Average**	**3-period Centered Moving Average**	**Exponential Smoothing α = 0.3**
MAE	11,284	15,928	4,702	7,798
MAPE	0.14	0.21	0.06	0.12
MSE	229,196,109	292,206,894	56,284,583	152,495,043
RMSE	15, 139	17,094	7,502	12,349
Theill's U	1.0	1.13	0.50	0.82

Table 3. Comparison of the four forecasting techniques.

As shown in Table 3, of the simple techniques employed, the 3-period centered moving average gives the best forecast since it has the lowest absolute errors and has a Theill's U value less than 0.55.

Applying both additive and multiplicative decompositions to the time series yielded the following results. For the additive model:

$$\hat{Y}_t = Tr_t + Sn_t$$

where

$$Tr_t = 72{,}995 + 38{,}624 \ x \ t$$

with seasonal estimates S_1 = 25,095, S_2 = - 8,624, S_3 = 46,139, and S_4 = 29,308. The value of Theill's U for this model is 0.29.

For the multiplicative model:

$$\hat{Y}_t = Tr_t \times Sn_t$$

where

$$Tr_t = 50{,}254 + 34{,}636 \ x \ t$$

with seasonal estimates S_1 = 1.2978, S_2 =1.1384, S_3 =0.9730, and S_4 = 1.3904. The value of Theill's U for this model is 0.36. Both forms of decomposition yield better Theill's U. Since the additive decomposition yields the best accuracy measure, this model can be used in forecasting the monthly future values for the usage for content services.

6 Research Directions

As more and more people get access to mobile devices, research on network traffic management experiences a steady increase in importance. Several different approaches for network usage modeling have been proposed. Kohandani et al [11] introduced a new forecasting technique called the extended structural which is based

on the basic structural model and to which new parameters have been added. Ikoma et al [12] used non-stationary Bayesian modeling to model network traffic of dial-up access in a university. This paper used traditional methods for modeling the usage characteristics of content providers. Future research on this area can attempt newer approaches to modeling. As the CP business matures and more data are gathered, the components of the time series may be updated to reflect the change in the environment. Hence, another direction that can be taken is to update the parameters of the estimation.

References

[1] J. Toral, "State of Wireless Technologies in the Philippines", *DigitalFilipino.com*.

[2] H. Elbiaze, O. Cherkaoui, B. McGibbon, and M. Blais, "A Structure-Preserving Method of Sampling Self-Similar Traffic", *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'05).*

[3] S. Pemmaraju, "Prediction of Demand/Usage Patterns for Services in Telecommunications Using Fuzzy Neural Networks", *IEEE*, 1997.

[4] O. Simula, J. Hollmen and E. Alhoniemi, "Models from data: analysis of industrial processes and telecommunication systems".

[5] P. N. Tan, M. Steinbach and V. Kumar, *Introduction to Data Mining*, Addison Wesley, 2006.

[6] P. Gaynor and R. Kirkpatrick, *Introduction to Time-Series Modeling and Forecasting in Business and Economics*, McGraw-Hill, 1994.

[7] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*, 2nd edition, Springer-Verlag, 2002.

[8] P. Damrongkulkamjorn and P. Churueang, "Monthly Energy Forecasting Using Decomposition Method with Application of Seasonal ARIMA", *Proceedings of The 7th International Power Engineering Conference 2005, IPEC 2005.*.

[9] G. Gray and P. J. Thomson, "On a family of finite moving average trend filters for the ends of series", *Journal of Forecasting*, 21, issue 2:125-149, 2002.

[10] G. W. Taylor, "Smooth transition exponential smoothing", *Journal of Forecasting*, 23, issue 6:385-404, 2004.

[11] F. Kohandani, D. W. McAvoy, and A. K. Khandani, "Wireless Airtime Traffic Estimation Using a State Space Model", *Proceedings of the 4th Annual Communication Networks and Services Research Conference (CNSR'06).*

[12] N. Ikoma, T. Shimizu, K. Imazu, and K. Yana, "Decomposition of Network Traffic of Dialup Access into Trend and Periodic Components Based on Nonstationary Bayesian Model", *Proceedings of the SICE Annual Conference, Fukui, 2003*.

Hierarchical approach to dynamics of criminality

D.V Serebryakov[1], I.V. Kuznetsov[2], M.V. Rodkin[2] *

1- Keldysh Institute for Applied Mathematics RAS – 17-th Dept
125047 Moscow, Miusskaya pl., 4 – Russia
2- International Institute of Earthquake Prediction Theory and Mathematical Geophysics RAS – Dept of Nonlinear dynamics
117556 Moscow, Warshavskoye sh., 79, korp. 2 – Russia

Abstract. This work is concerned to criminality forecast methods based on universal behavior rules of complex nonlinear hierarchical systems. In the work we apply an approach used in earthquake prediction methods, namely, a special reoccurrence graph slopping behavior prior to a flash-up. Using a weekly criminality militia data from Yaroslavl we study dynamics of criminality and propose forecast algorithms for serious crime flash-up, which forecasting about 70% of flash-ups. All alarms occupy about 30% of whole time, and false alarms covered 8% of the alarm time. The meantime of alarm makes up about 3 weeks.

1 Introduction

In this work we apply methodology using in earthquake prediction theory where one should predict a system state flash-up. This way is based on results of synergetics – the science investigating universal rules of complex dynamic systems' behavior, which are poorly amenable to routine modeling. It is convenient to consider this kind of systems in frames of a hierarchical nonlinear model. The one of these systems' main characteristic is capability to selforganization that sometimes results in abrupt system behavior changes, as changes of external conditions are small.

Nonlinear hierarchical systems aren't reduced to a simple sum of its components. An interconnection of these parts plays the most important role there, which results in changing in properties of the system in whole. It is suggested that processes on any system's hierarchical level have an identical structure with each other (dimensioned invariance, self-similarity).

If a model of the system is correct, has a set of necessary prognosis properties (correctly selected phase space) and enables to make a well-quality stable prediction then one may make the prediction more accurate by considering the system process on lower hierarchical levels at time and space points where present alarm has place.

* This work was supported by the EC Project "Extreme events: Causes and consequences (E2-C2), Contract No 12975 (NEST)", Russian for Basic Research (project #04-01-00510), Russian Humanitarian Scientific Foundation (project #05-03-03188).

It is this way [1-6] that enabled to create algorithms predicting heavy earthquakes [7,8], senate and president election results in USA [9,10], economic recessions and unemployment in USA and Europe and economic crisis in Russia [11-12].

An investigating of Yaroslavl crime time series as one of important society state characteristics resulted in developing of serious crime flash-up forecast algorithms that we present in this article.

2 Data

In this work we use weekly crime data from Yaroslavl town, the administrative center of Yaroslavskaya oblast (region) in Russia, for time period from 9.02.1993 till 19.06.2001, we defined a beginning of a week as Tuesday. The monitoring time is 437 weeks, i.e. there are 437 cases or observations in initial time series. There are 35 types of crime in militia list for the period (see Tabl.2).

3 Algorithm descriptions

3.1 Heart of the methods

In general, posing of the problem is following.

We want to construct a predictor function F that for every time moment t must indicate in dependence on preceding data if the next time interval from $t + 1$ to $t + d$ contains ($F = 1$) or doesn't contain ($F = 0$) a serious crime flash-up.

The following modeling is based on our supposition that arising and preparation of serious crime flash-up are similar to progress of powerful events in hierarchical systems, which, as a rule, is prepared by preliminary activation on previous hierarchical levels of such systems. This scheme takes place in processes of powerful earthquakes' preparation and crack formation by external strain acting in model destruction experiments. In such processes we may define *a power of event* (earthquake, crack, crack arising etc.) in a system. Power for earthquake is its energy, for crack is its dimension. Values of power arrange events of the systems by hierarchical levels of such system. The larger power value of an event the higher a hierarchical level for this event.

It is known earthquake number N distributions by its energy (power) P or cracks by its dimension have power-mode nature for a long enough time interval. Such distributions represented in log-log scale have linear type characterized by its slopping b:

$$\lg N = a - b \lg P. \qquad (1)$$

In seismology this is named as the Gutenberg-Richter reoccurrence law [13], graphically it is represented by reoccurrence graph. Logarithm of earthquake energy is named *a magnitude* of this earthquake.

We construct an algorithm of serious crime flash-up prediction by behavior of b's analogue for crime statistic. Applying this scheme for crimes we expect an increase of less hazardous crimes prior to a flash-up of more hazardous crimes.

Adaptation of methods developed to predict physical phenomena for social ones is complicated by absence of event power metering scale. Therefore the next main step in the model creation is to produce a power scale for crimes.

3.2 Ranking manner

Rank	Classification
G3	01-homicide (895).
G2	02-aggravated assault with serious body injuries (1976), 04-forcible rape (761), 05-dangerous armed robbery (1058), 06-robberies (10044), 18-burning (348).
G1	03- assault with body injuries (24457), 07-other assault and hooliganism (4504), 08-burglaries (60909), 09-embezzlement (86), 10-bribery (25), 11-coinage offence (1525), 12-escape (19), 13-motor vehicle theft (2593), 14-resistance to militia (534), 15-extortion (902), 16-dissolute actions (126), 17-frauds (2621), 30-other crimes (7851), 39-finding of a body (5756), 41-unknown disappearance of people (5078).

Tabl.1. The first ranking manner of crime types; "01", "02" and so on are numeric marking of crime types, in parentheses – number of crimes for the monitoring period.

Rank	Classification
G5	01-homicide (895), 02-aggravated assault with serious body injuries (1976), 04-forcible rape (761), 05-dangerous armed robbery (1058), 41-unknown disappearance of people (5078).
G4	06-robberies (10044), 38-suicide (1929), 39-finding of a body (5756).
G3	03-assault with body injuries (24457), 09-embezzlement (86), 10-bribery (25), 14-resistance to militia (534), 15-extortion (902), 17-frauds (2621), 18-burning (348), 32-air crash (0), 33-railway accident, 42-lost-the-way, thrown children (10), 44-mass poisoning of people (0), 45-application and use of the organic weapon (9).
G2	07-other assault and hooliganism (4504), 08-burglaries (60909), 11-coinage offence (1525), 16-dissolute actions (126), 40-infection with poisonous and chemical substances (1), 43-mass disorders.
G1	12-escape (19), 13-motor vehicle thefts (2593), 30-other crimes (7851), 34-road accident (4571), 35-sudden death (6903), 36-fire (2345), 37-accident (2264), 60-other incident (20755), 61-accomplished earlier crime solving (26).

Tabl.2. The second ranking manner of crime types (consists all types we obtained); "01", "02" and so on are numeric marking of crime types, in parentheses – number of crimes for the monitoring period (absence of parentheses means there wasn't data).

As our algorithm is based on behavior of b calculated by using logarithm of event power it is more convenient to characterize a crime by quantity that is analogue to $\lg P$ in (1). Value of this quantity indicates number of group in crime type hazard hierarchy. We name this quantity as *crime power*. Consequently crime power values are

discrete and limited by number of groups. Totality of the groups and its interconnections are the system describing crime dynamics.

Here we use an expert estimation to measure power of a crime type. Furthermore, to obtain stable and statistically representative results it is advisable to rank all crime types to small number of groups. In the work we use two ranking manner: the first ranking manner – for 3 groups (see Tabl.1), the second ranking – for 5 groups (see Tabl.2). Events of a crime type having the highest group number (3 for the first ranking and 5 for the second one) are named as *serious crimes*.

In addition there are two ways to produce reoccurrence graph for one ranking manner: *non-cumulative* (original), as represented in the tables, and *cumulative*. For instance, the 1st cumulative group *K1* for the first ranking consists on original groups *G1*, *G2*, *G3*, the 2nd group *K2* consists on non-cumulative groups *G2* and *G3*, and the 3d one *K3* coincides with *G3*. Reoccurrence graphs and corresponding data by groups for the monitoring period, which are approximated by the graphs, are shown in Fig.1 (for the first ranking *b*'s equal 1,0429 and 1,0710 for original and cumulative way correspondingly, for the second ranking *b*'s equal 0,1944 and 0,3136 for original and cumulative way correspondingly).

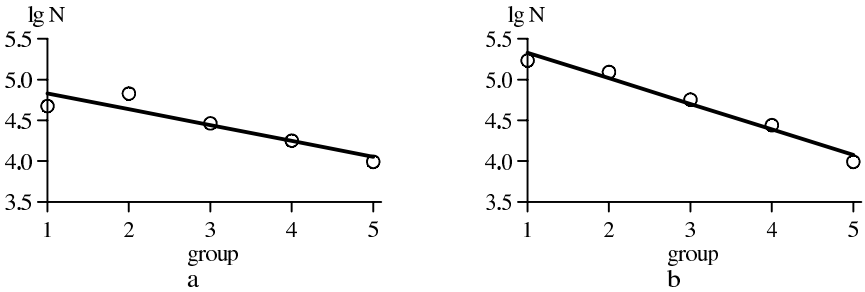

Fig. 1: The reoccurrence graphs and corresponding them data (circles) for the second grouping for the monitoring period: a) non-cumulative, $b = 0{,}1944$; b) cumulative, $b = 0{,}3136$.

3.3 Object-to-predict

Object or *object-to-predict* is a time moment where a flash-up of the serious crimes arises plus a special condition. We believe it is advisable to determine a flash-up as an overshoot of present crimes' number comparatively to mean number of crimes for a previous time interval. We say that there is *a flash-up* at the time moment *i*, if satisfied (method *S*)

$$R_i = N_i - n_i \geq \sigma \tag{2a}$$

or (method *D*)

$$R_i = N_i / n_i \geq \sigma, \tag{2b}$$

where N_i is number of present serious crime events for the time moment *i*, the mean number of serious crime events for the previous time interval from *i*-5 to *i*-1 is equal to

$n_i = (N_{i-1} + N_{i-2} + N_{i-3} + N_{i-4} + N_{i-5}) / 5$, σ is an *intercepting threshold*, we name set of R_i's as *a residue series*.

It is possible when two or more flash-up are close to each other. In earthquake prediction theory such flash-ups following the first one is named aftershock and doesn't considered as object when the first flash-up is in fact the object. We think two different flash-ups at moments i and j, $i < j$ fix two different objects if $j - i > 2$, in opposite case only moment i is an object. This is the special condition mentioned above.

3.4 Predictor functions

Prior to the objects we wait an increase of b. We believe to observe such increase it is sufficient to consider time interval consisting about 5 observations.

For the predictor function $F1(i)$ we define increase of b_i by analogy with flash-up defining for objects. $F1(i)$ equals 1 if satisfied

$$b_i - k_i \geq \beta \tag{3a}$$

where b_i is slopping for the time moment i, $k_i = (b_{i-1} + b_{i-2} + b_{i-3} + b_{i-4} + b_{i-5}) / 5$ is mean slopping for the 5 previous moments, β is *an intercepting threshold*.

To calculate slopping b_i we construct reoccurrence graph, the line $y_j = a_i - b_i j$. This line approximates for the first grouping the set $\{(N_1, 1), (N_2, 2), (N_3, 3)\}$, where N_1, N_2 and N_3 are sum of numbers of events for crime groups *G1*, *G2*, *G3* correspondingly at the time moments i-4, i-3, i-2, i-1, i; and for the second ranking the set $\{(N_3, 3), (N_4, 4), (N_5, 5)\}$, where N_3, N_4 and N_5 are sum of numbers of events for crime groups *G3*, *G4*, *G5* correspondingly at the time moments i-4, i-3, i-2, i-1, i.

For the predictor function $F2(i)$ we define increase of b_i as serial increase of b's values over the 5 time moments $i_5 < i_4 < i_3 < i_2 < i_1$, $i_1 \equiv i$ in condition that $i_j - i_{j+1} \leq 3$ and $b(i_{j+1}) \geq b_m \leq b(i_j)$ if $i_{j+1} < m < i_j$, where $j = 1, 2, 3, 4, 5$. $F2(i)$ equals 1 if satisfied

$$b(i_5) < b(i_4) < b(i_3) < b(i_2) < b(i) \tag{3b}$$

where $b(i) \equiv b_i$ is slopping calculated as for the *F1* but for all groups in given ranking.

We say there is *a predictor signal* at the time moment i when using one of the predictor function if this function's value equals 1.

3.5 Alarm

If a predictor signal presents at time moment i, then one *declares an alarm* for following d serial moments i+1, i+2,…, i+d, i.e. one should wait for an object appearance over these moments named as *alarm interval* or *alarm*. If there is other predictor signal at the moment j during the alarm interval, then the alarm is prolonged for the next d moments j+1, j+2,…, j+d. If there is an object s during the alarm, then the final alarm moment is s after which this alarm is cancelled. If an object s and a predictor signal i present at the same moment or $0 \leq i - s \leq 2$, then an alarm is not declared: two

serial moments followed an object are considered as relaxation period when system behavior is special and isn't applicable for the prediction.

We say that *an alarm is successful* if the alarm consists an object-to-predict, *an alarm is false* (*false alarm*) if there is no an object over the alarm interval.

We say that *the object is predicted* if it is in an alarm, an object-to-predict is *fail to predict* (*a fail-to-predict object*) if it is not covered any alarm interval.

3.6 Examples of algorithms prediction

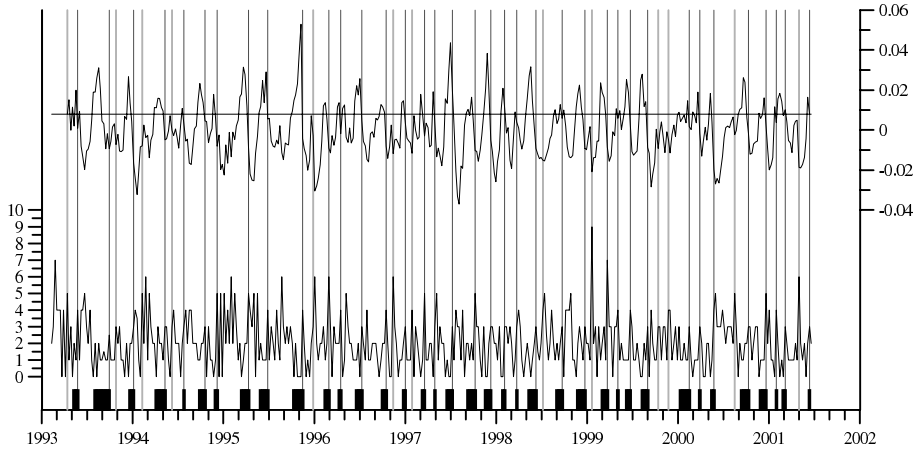

Fig. 2: Prediction results. Predictor function *F1*, the first non-cumulative ranking, the method *S*, $\sigma = 1{,}4$; $\beta = 0{,}008$; $d = 6$. Lower graph – original series *G3* (left vertical axis); upper graph – residue series (right vertical axis), the intercepting threshold β (horizontal line); grey vertical lines – fail-to-predict objects, black vertical lines – predicted objects; rectangles – alarms.

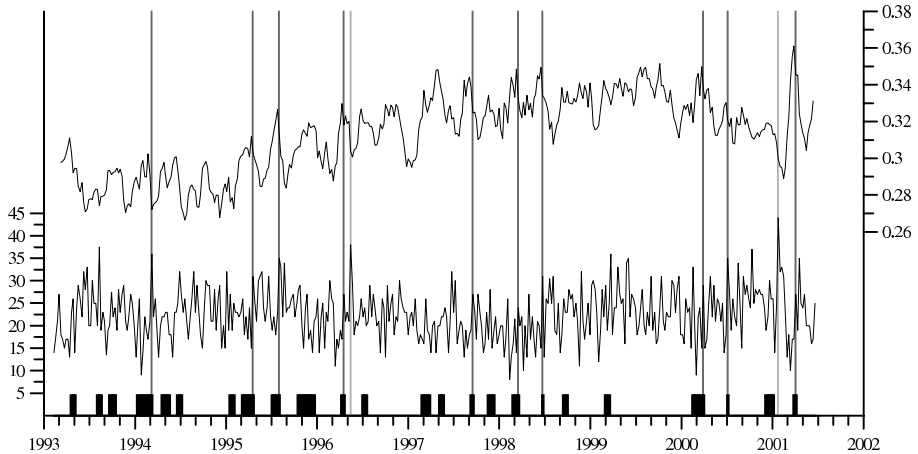

Fig. 3: Prediction results. Predictor function *F2*, the second cumulative ranking, the method *D*, $\sigma = 1{,}6$; $d = 3$. Lower graph – original series *G5* (left vertical axis); upper graph – slopping series (right vertical axis); grey vertical lines – fail-to-predict objects, black vertical lines – predicted objects; rectangles – alarms.

Figure 2 shows graphically results of prediction algorithm using the method *S* for object defining and the predictor function *F1* for the first non-cumulative ranking as $\sigma = 1{,}4$; $\beta = 0{,}008$; $d = 6$. For this case we predicted 37 of 50 objects (74%) as duration of all alarms is 125 of 428 weeks that consists 29,2% of the monitoring time as no one false alarm.

Figure 3 shows graphically results of prediction algorithm using the way *D* object defining and the predictor function *F2* for the 2nd cumulative ranking as $\sigma = 1{,}6$; $d = 3$. For this case we predicted 10 of 12 objects (83%) as time alarm is 22,7% of all monitoring time, but false alarm time is 58,8% of the monitoring time, that is why we may not say this result is well.

4 Numerical experiments by prediction algorithm testing

Thus we suggest the following parameters of serious crime flash-up forecast:

- two crime type grouping manner;
- two variants of object-to-predict defining – by residues *S* and ratio *D* with intercepting threshold σ;
- two variants to produce reoccurrence graph – non-cumulative *nK* and cumulative *K*;
- two prediction functions – *F1* with intercepting threshold β and *F2*;
- alarm duration *d*.

Let's denote total objects-to-predict number as *N*, predicted objects as N^+, then number of fail-to-predict objects is $N^- = N - N^+$. Prediction algorithm quality is convenient to evaluate by a η-τ diagram [8]: $\eta = N^- / N$, and $\tau = T_a / T$, where T_a is total alarms duration. As $\eta = 1$, $\tau = 0$ we have one extreme case when all objects are fail-to-predict as time alarm equal zero ("strategy of optimist"). As $\eta = 0$, $\tau = 1$ we predict all objects when time alarm is all monitoring period ("strategy of pessimist").

An integral quality of prediction when free algorithm parameters are changed might be evaluated by quantity $\varepsilon = \min(\eta+\tau)$. Condition $\eta+\tau < 1$ ($\varepsilon < 1$) corresponds to a non-trivial prediction. The ε is smaller the prediction is better. For earthquake prediction ε is located in interval 0,32-0,5. For examples above ε's equal 0,427 and 0,516. Tabl.3 represents a few other results of forecast algorithms.

Results of algorithm testing help us to outline algorithm parameter interdependence. Thus the σ is smaller the number of objects is greater, but there is a lower limit for σ. When σ is smaller it the object-to-predict number doesn't increase: intervals between objects comes to its minimum – 3 weeks. It is one limitation of suggested algorithm: it is not able to predict too small flash-ups.

Also there is a lower limit for β. When β is smaller it then alarm duration becomes more 30% of all monitoring time that is not convenient for practically use.

The results we obtained (some of them are represented in Tabl.3 and shown in Fig.4) are quite well. We can see that we may consider as a predictor an increase of reoccurrence graph slopping *b*.

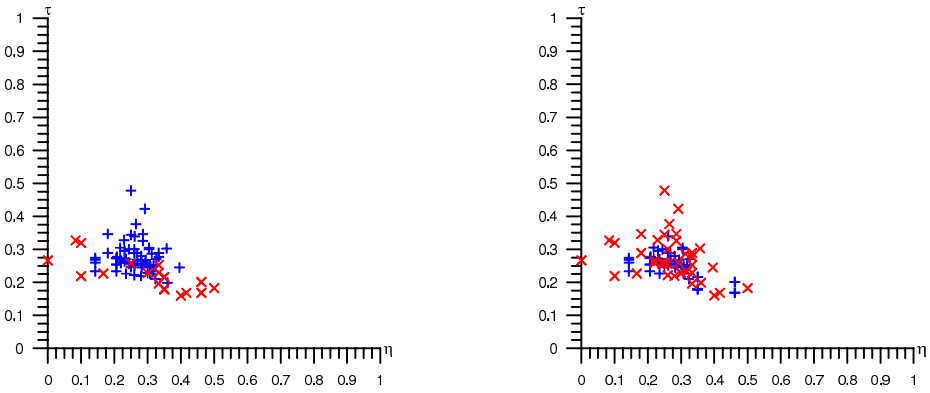

Fig. 4: Comparison of forecast algorithm for different parameters:
a) predictor function: "+" – *F1*, "×" – *F2*; b) ranking manner: "+" – 1st, "×" – 2nd

S/D	*nK/K*	σ	β	*d*	N^+:*N*	η	τ	ε	*F*	*Tf*/T_a
the first grouping										
S	nK	1,4	0,008	6	37:50	,260	,292	,552	1	0
S	nK	1,6	0,008	3	32:46	,304	,304	,608	1	0,277
S	nK	1,6	0,008	6	34:46	,261	,343	,604	1	0,129
S	nK	1,6	0,010	3	31:46	,326	,273	,599	1	0,274
S	nK	1,6	0,010	6	32:46	,304	,308	,612	1	0,220
D	nK	2,4	0,011	3	27:34	,206	,257	,463	1	0,327
D	nK	2,4	0,012	3	26:34	,235	,229	,464	1	0,337
D	nK	2,4	0,012	6	26:34	,235	,273	,508	1	0,333
D	K	2,4	0,011	3	27:34	,206	,257	,463	1	0,336
D	K	2,4	0,012	3	27:34	,206	,236	,442	1	0,297
D	K	2,4	0,012	6	27:34	,206	,278	,484	1	0,327
D	K	2,4	0,013	3	23:34	,324	,213	,537	1	0,396
D	nK	2,6	-	5	14:26	,462	,201	,662	2	0,523
D	K	2,6	-	3	14:26	,462	,168	,630	2	0,458
the second grouping										
D	nK	1,1	0,005	3	37:50	,260	,304	,564	1	0,246
D	nK	1,1	0,010	3	35:50	,300	,231	,531	1	0,232
D	nK	1,1	0,010	6	37:50	,260	,259	,519	1	0,126
D	nK	1,2	0,010	3	32:48	,333	,292	,625	1	0,208
D	K	1,1	0,010	3	37:50	,260	,224	,484	1	0,250
D	K	1,1	0,010	6	39:50	,220	,262	,482	1	0,196
D	K	1,1	0,005	3	41:50	,180	,292	,472	1	0,240
D	K	1,2	0,010	3	33:48	,313	,290	,603	1	0,306
D	nK	1,3	0,015	3	27:42	,357	,306	,663	1	0,305
D	K	1,5	-	3	16:24	,333	,196	,530	2	0,429
D	K	1,6	-	3	10:12	,200	,227	,427	2	0,588
D	K	1,9	-	3	2:2	0	,266	,266	2	0,886

Tabl.3. Results of some numeric experiments of prediction algorithm. *Tf* is false alarm time.

5 Conclusions

We showed [14] that regime of crime number dynamic is analogous in many respects to behavior of complex systems with precursor activation effects before serious crime flash-ups. The phenomenon of activation appears as less hazardous crime "germination" to more hazardous ones.

By analogy with a model of extreme powerful event progress [15] serious crime realization might be treated as an assembly of unstable system state creation effects and some trigger mechanism. This trigger may consist in abrupt nature factors changing and even additional drink of alcohol.

Represented in the work the serious crime prediction methods could be used in tactical controlling of emergency town services, for instance, police, ambulance, hospitals etc. Having information about possible serious crime flash-up over the next few weeks authorities could reinforce such services what allows to guarantee their faster reaction upon crime events. Decreasing of time reaction upon crime will lead to decreasing of deceases and dangerous health hazards as a result of violent acts. Such information could help to effectively control working regime of emergency services' staff: vacations, free days, duties etc.

Knowing us works in crime forecasting use other methods. Most part of them [16-18] aimed to forecast the next number of crime events applying extrapolation methods. In spite of vast using and detailed developmental work of such methods they at present have similar forecasting quality [19] and limitations, in particularly, number of events should be several tens. In addition, extrapolation methods cannot, in principal, predict flash-ups of forecasting quantity; moreover, flash-up time moments are excluded for forecast method quality evaluation. This makes impossible to compare predictability of extrapolation methods and our ones that aimed to predict just flash-up time moments.

Revealed patterns in crimes regime and developed serious crime flash-up forecast algorithms might be used further as components of social tension level monitoring system for society. This work concerns questions of crime regime "self-progress" as one of society behavior aspects. To create more comprehensive monitoring system based on universal complex nonlinear hierarchical system behavior patterns revealed in frame of synergetics it is necessarily to add components that take into account influence of social-economics, man-caused and natural factors.

References

[1] G. Nicolis and I. Prigozhin. Selforganization in non-equilibrium systems. Moscow, Mir, 1979.

[2] G. Haken, Synergetics. Hierarchy in selforganizing systems and devices. Moscow, Mir, 1985.

[3] P. Bak and C Tang. Earthquake as a self-organized critical phenomenon. J. Geophys. Res., pages 15635-15637, V.94, 1989.

[4] V.I. Arnold. Theory of catastrophe. Moscow, Nauka, 1990.

[5] E.N. Lorenz. The essence of chaos. London: U.C.L. Press Ltd., 1993.

[6] D.L. Turcotte. Chaos, fractals, nonlinear phenomena on Earth sciences. U.S. National Report to IUGG 1991-1994, Rev. of Geophis. supplement. AGU, pagers 341-343, 1995.

[7] V.I. Keilis-Borok, eds. Intermediate-term earthquake prediction: models, algorithms, worldwide tests. Phys. Earth Planet. Inter. V.61, N1-2 (Spec. Iss.), 1990.

[8] V.I. Keilis-Borok, A. Soloviev, eds. Nonlinear Dynamics of the Lithosphere and Earthquake Prediction. Springer, Heidelberg, 1-36, 2003.

[9] A. Lichtman, V.I. Keilis-Borik. Pattern recognition applied to presidential elections in the United States 1860-1980; Role of integral social, economic and political traits// Proc. Nat. Acad. Sci. USA. V.78, pagers 7230-7234, 1981.

[10] A. Lichtman, V.I. Keilis-Borik. Aggrigate-level analysis and prediction of midterm senatorial election in the United States, 1974-1986// Proc. Nat. Acad. Sci. USA. V.86, pagers 10176-10180, 1989.

[11] V. Keilis-Borok , J.H. Stock, A. Soloviev, P. Mikhalev. Pre-recession pattern of six economic indicators in the U.S.A., Journal of Forecasting, V.19, pagers 65-80, 2000.

[12] I. Kuznetsov, E. Grebenuk, D. Muratov. About forecasting of crisis events. Poceeding of International conference "Mathematical modeling of social and economics dynamics", Moscow, pagers 174-177, 2004.

[13] B. Gutenberg and C. F. Richter. Earthquake magnitude, intensity, energy and acceleration. Bull. Seismol. Soc. Am., 46, pagers 105–145, 1956.

[14] D.V. Serebryakov, I.V. Kuznetsov, M.V. Rodkin. A serious crime flash-up forecast based on hierarchy of criminality behavior. Preprint, Inst. Appl. Math., the Russian Academy of Science, number 15, 2005. http://www.keldysh.ru/papers/2005/prep12/prep2005_12.html

[15] M.V. Rodkin. Model of synergetical effect progress in porewful catastrophes. Geoecology, number 1, pagers 1-7, 2005.

[16] W. Gorr, R. Harries. Introduction to crime forecasting. International Journal of Forecasting, V 19, pagers 551-555, 2003.

[17] R. Harries. Modelling and predicting recorded property crime trends in England and Wales — A retrospective. International Journal of Forecasting, V 19, pagers 557-566, 2003.

[18] S.F.H. Allison, A.M. Schuck, K.M. Lersch. Exploring the crime of identity theft: Prevalence, clearance rates, and victim/offender characteristics. Journal of Criminal Justice, V. 33, pagers 19-29, 2005.

[19] J.S. Armstrong. Forecasting by extrapolarion: Conclusions from 25 years of research. Interfaces, V. 14, pagers 52-66, 1984.

Exponential smoothing methods for interval time series

Javier Arroyo[1], Antonio Muñoz San Roque[2], Carlos Maté[2], and Ángel Sarabia[2] *

1- Universidad Complutense - Facultad de Informática
Profesor José García-Santesmases s/n, 28040 Madrid - Spain

2- Universidad Pontificia Comillas - Instituto de Investigación Tecnológica
Alberto Aguilera 25, 28015 Madrid - Spain

Abstract. An interval time series is a sequence of intervals observed sequentially in time. It allows to describe the behavior of phenomena where variability must be taken into account. In this paper, exponential smoothing methods are adapted to this kind of series with the help of interval arithmetic. A comparison of the forecast performance of the interval exponential smoothing and other methods is carried out. These methods include interval multilayer perceptron and modelling, in a separate way, the series of the interval attributes (minimum and maximum, or center and radius) using classical forecasting methods.

1 Introduction

An interval time series (ITS) is a sequence of intervals observed sequentially in time. ITS has been previously proposed in [1] and [2] with the aim to extend symbolic data analysis [3] to the field of time series. In symbolic data analysis, items are described by symbolic variables: lists of values, intervals, frequency distributions, etc. These variables allow the characterization of complex real-life situations and the summarization of large data sets into symbolic ones retaining the key information but offering a more manageable size. In an ITS, the variable observed through time is an interval variable.

ITS represent phenomena that classical time series (i.e. series where observations are single values) cannot accurately describe, such as when variability must be taken into account. For example, an ITS is suitable to describe the lower and upper monthly water levels of a river at a given location; or the range of daily values of a stock index; or the intervals enclosing the levels of an air-pollutant recorded in several meteorological stations distributed along a city.

ITS can be obtained in sampling or summarization contexts. In a sampling context, an ITS arises recording the lower and upper values in each time interval. In a summarization context, an ITS is obtained summing up a set of values by means of an interval for each considered instant. In these contexts, intervals can arise from the minimum and maximum observed values, but also from the interquartile range or from the middle 90% of the scores (in order to avoid outliers); it depends on the aims of the analysis.

*This work is funded by the Dirección General de Universidades e Investigación of Madrid, by the Universidad Complutense (Research Group 910494), and by Universidad Pontificia Comillas (PRESIM project).

This paper tackle ITS forecasting from different approaches. Section 2 defines interval variable and ITS. Section 3 summarizes the main ideas of interval arithmetic, which will be used as the basis in some ITS forecasting methods. Section 4 briefly shows how to measure errors in ITS. In section 5, exponential smoothing methods for ITS are proposed and some ideas to deal with trend and seasonality are introduced. Section 6 shows other approaches to forecast ITS and special attention is given to the Interval Multilayer Perceptron. In section 7 the forecasting performance of the proposed approaches is analyzed by an example. Finally, section 8 concludes.

2 Definitions

An interval variable, $[X]$, is a variable defined for all the elements i of a set E, where $[X]_i = \{[X_{i,L}, X_{i,U}], -\infty < X_{i,L} \leq X_{i,U} < \infty\}, \forall i \in E$. The value of $[X]$ for the ith element can be denoted by the interval lower and upper bounds $[X]_i = [X_{i,L}, X_{i,U}]$ or, equivalently, by the center and radius $[X]_i = \langle X_{i,C}, X_{i,R} \rangle$, where $X_{i,C} = (X_{i,L} + X_{i,U})/2$ and $X_{i,R} = (X_{i,U} - X_{i,L})/2$, respectively.

An ITS can be denoted by $\{[X]_t\}$ and the value of the variable in t can be expressed as $[X]_t = [X_{t,L}, X_{t,U}] = \langle X_{t,C}, X_{t,R} \rangle$. In order to denote a forecasted value, a hat will be placed above the variable, $[\hat{X}]_t$.

3 Interval arithmetic

Apart from symbolic data analysis, other field related with intervals is interval analysis [4]. This field assumes that, in the real world, observations and estimations are usually incomplete or uncertain. Thus, it considers that, if precision is needed, data must be represented as intervals enclosing the real quantities. The theory of interval analysis is build around this idea.

Interval computations are based in interval arithmetic [4], which can be summarized as follows: Let A and B be two intervals and $\Box$ be an arithmetic operator, then $A \Box B$ is the smallest interval which contains $a \Box b \; \forall a \in A$ and $\forall b \in B$. According to this definition, interval addition, subtraction, multiplication and quotient are respectively, defined by:

$$[A] + [B] = [A_L + B_L, A_U + B_U]$$

$$[A] - [B] = [A_L - B_U, A_U - B_L]$$

$$\begin{aligned} [A] \cdot [B] \quad = \quad & [\min\{A_L \cdot B_L, A_L \cdot B_U, A_U \cdot B_L, A_U \cdot B_U\}, \\ & \max\{A_L \cdot B_L, A_L \cdot B_U, A_U \cdot B_L, A_U \cdot B_U\}] \end{aligned}$$

$$[A]/[B] = [A] \cdot (1/[B]), \text{with } 1/[B] = [1/B_U, 1/B_L]$$

It is worth noting that interval arithmetic subsumes the classical one, in the sense that, if the operands of interval arithmetic are intervals with width zero

(i.e. $[a,a], a \in \Re$), the result of interval operations will be equal to the result obtained by the classical operations.

In interval arithmetic, addition and multiplication are associative and commutative. The distributive law does now always hold, but the subdistributive property is satisfied; it is defined as:

$$[A]([B]+[C]) \subseteq [A][B]+[A][C]$$

4 Error Measures for Interval Time Series

According to [2], error measures for ITS cannot be based in the difference between the observed and the actual interval, because interval subtraction does not faithfully represent the concept of deviation, as $[A]-[A]=[0,0]$ if and only if $[A]=[a,a]$ with $a \in \Re$. Thus, they propose error measures based on distances for interval data, such as the Hausdorff and the Ichino-Yaguchi distance.

Let $\{[X]_t\}$ be the observed ITS, and $\{[\hat{X}]_t\}$ be the forecast of this ITS with $t=1,...,n$, the Mean Distance Error based on Hausdorff distance is defined as

$$MDE_H = \frac{1}{n}\sum_{t=1}^{n}[|X_{t,C}-\hat{X}_{t,C}|+|X_{t,R}-\hat{X}_{t,R}|],$$

and the Mean Distance Error based on the Ichino-Yaguchi distance is defined as

$$MDE_{IY} = \frac{1}{n}\sum_{t=1}^{n}0.5[|X_{t,L}-\hat{X}_{t,L}|+|X_{t,U}-\hat{X}_{t,U}|].$$

5 Exponential Smoothing methods for ITS

Exponential smoothing methods in classical time series obtain forecasts as the weighted moving average of all past observations where the assigned weights decrease exponentially (see [5] for an up-to-date review). In this section, exponential smoothing methods are adapted to ITS[1]. In order to adapt the methods to ITS, a procedure to average intervals is required. We propose to average intervals using interval arithmetic.

5.1 Average Interval

The interval that averages a set E of n intervals $[X]_i$, $i=1,...,n$ is defined as

$$[\bar{X}] = \frac{[X]_1+[X]_2+...+[X]_n}{n},$$

where the arithmetic operations are interval arithmetic operations.

The average interval holds the following properties:

$$\bar{X}_L = \frac{X_{L,1}+X_{L,2}+...+X_{L,n}}{n}, \quad \bar{X}_U = \frac{X_{U,1}+X_{U,2}+...+X_{U,n}}{n},$$

[1]The notation of the proposed methods will be similar to that in [5]

$$\bar{X}_C = \frac{X_{C,1} + X_{C,2} + ... + X_{C,n}}{n},\ \bar{X}_R = \frac{X_{R,1} + X_{R,2} + ... + X_{R,n}}{n}.$$

These properties allow us to consider that the average interval is the barycenter of a system of particles (the set of intervals E) where each particle is defined by two coordinates: the lower and the upper bounds, or, equivalently, the center and the radius. Obviously, as the average interval is equivalent to the interval barycenter, it can also be seen as the interval that minimizes the addition of the euclidean distances between itself and each interval of the set E.

The definition of moving averages based in the interval average is straightforward, and will not be tackled in this article.

5.2 Simple Exponential Smoothing

The formula of simple exponential smoothing (SES) in classical time series is

$$\hat{X}_{t+1} = \hat{X}_t + \alpha(X_t - \hat{X}_t),$$

where $\alpha \in [0, 1]$. This equation is written in an error-correction form, while its equivalent recurrence form is given by:

$$\hat{X}_{t+1} = \alpha X_t + (1 - \alpha)\hat{X}_t.$$

If both equations are adapted to ITS using interval arithmetic, they are not equivalent and

$$\alpha[X_t] + (1 - \alpha)[\hat{X}]_t \subseteq [\hat{X}]_t + \alpha([X_t] - [\hat{X}_t])$$

due to the subdistributive property. Thus, the SES method for ITS will adapt equation in recurrence form as it produces tighter intervals. It is defined as:

$$[\hat{X}]_{t+1} = \alpha[X_t] + (1 - \alpha)[\hat{X}]_t,$$

where $\alpha \in [0, 1]$. The initializing phase requires the value of $[\hat{X}]_1$ which can be the first observed value, $[X]_1$, or the average interval of the first three or four observed values. It is clear, that the SES forecast of an ITS is the weighted moving average of all past observations, where the weights decrease exponentially.

5.3 Exponential Smoothing with Trend

The Holt exponential smoothing method allows forecasting classical time series with trend. This method smoothes both the level and the trend of the series. The adaptation of the Holt method for ITS requires to smooth both components.

In our approach, the level of the HTS in t will be represented by an interval, $[S]_t$; and the trend of the HTS in t will be represented by a single value, T_t, that represents the location of the interval by its center. Both, level and trend, are separately smoothed and later added in order to obtain the forecast.

The exponential smoothing method with additive trend (EST) is defined as:

$$[S]_t = \alpha[X]_t + (1 - \alpha)([S]_{t-1} + T_{t-1}),$$

$$T_t = \gamma(S_{C,t} - S_{C,t-1}) + (1-\gamma)T_{t-1},$$

$$[\hat{X}]_{t+m} = [S]_t + mT_t,$$

where $\alpha, \gamma \in [0,1]$, $S_{C,t}$ is the center of the interval $[S]_t$, and m is a factor that multiplies the trend in order to produce forecasts for m periods ahead. The initialization values can be $T_1 = X_{C,2} - X_{C,1}$ and $[S]_1 = [X]_1$, but more sophisticated initialization values can be given.

5.4 Exponential Smoothing with Seasonality

In an ITS, it can be considered that there are two different types of seasonality: first, considering that the seasonal variation only concerns to the location of the intervals; and second, considering that the seasonality affects the whole interval. We will propose two different methods in order to deal with both alternatives.

In the first approach, the seasonal component in t is represented as a single value, I_t, representing the changes in the interval location due to the seasonal effect and where interval location is represented as the interval center. The level in t, $[S]_t$, is a deseasonalized interval, i.e. an interval without the seasonal effect. Forecasts are the addition of the level interval and the seasonal component. The interval exponential smoothing method with additive crisp seasonality (IEScS) is given next:

$$[S_t] = \alpha([X]_t - I_{t-s}) + (1-\alpha)[S]_{t-1},$$

$$I_t = \delta(X_{c,t} - S_{c,t}) + (1-\delta)I_{t-s},$$

$$[\hat{X}]_{t+1} = [S]_t + I_{t-s+1},$$

where $\alpha, \delta \in [0,1]$ and s is the length of the seasonality. The initialization of the model requires a whole season (i.e. the first s periods) and it can be done as follows: $[S]_s = \frac{[X]_1 + ... + [X]_s}{s}$ and $I_1 = X_{C,1} - S_{C,s}$, $I_2 = X_{C,2} - S_{C,s}$,..., $I_s = X_{C,s} - S_{C,s}$.

In the second approach, the level, $[S]_t$, and the seasonality, $[I]_t$, are represented as intervals. In this case, the level is no deseasonalized and both components are independently smoothed in separate equations. The forecast is obtained as the weighted addition of the level and the seasonality interval, where the weight controls the importance of each component. The interval exponential smoothing method with additive interval seasonality (IESiS) is shown next:

$$[S]_t = \alpha[X]_t + (1-\alpha)[S]_{t-1},$$

$$[I]_t = \delta[I]_t + (1-\delta)[I]_{t-s},$$

$$[\hat{X}]_{t+1} = \xi[S]_t + (1-\xi)[I]_{t-s+1},$$

where $\alpha, \delta, \xi \in [0,1]$, and s is the length of the seasonality. As in the previous approach, the first s periods of the HTS are needed to initialize the model. The initialization of the seasonal component can be done as follows $[I]_1 = [X]_1$, $[I]_2 = [X]_2$, ..., $[I]_s = [X]_s$; while the initial value of the level component can be obtained as $[S]_1 = [X]_1$ for $t = 1$ and applying the smoothing equation to obtain the level for $t = 2, .., s$.

6 Other approaches to forecast ITS

The simplest way to forecast an ITS is the naive model, although its forecasting ability is limited. Letting aside this model, the most straightforward approach consists of transforming the ITS in a pair of classical time series (center and radius, or minimum and maximum) and modelling each of them with an univariate forecasting model or both of them with a multivariate model. This approach has the objection that it can produce wrong intervals (e.g. intervals where $X_L > X_U$ or where $X_C < 0$) as it does not deal with intervals as a whole. On the other hand, interval attributes allow to focus in the features that characterizes the intervals. We consider that the center-radius approach is especially interesting as the center shows the interval location and the radius shows the interval span.

Classical multilayer perceptrons are commonly applied in time series forecasting prediction [6]. In a similar way, the Interval Multilayer Perceptron (iMLP) [7] can be applied to forecast ITS. The iMLP adapts the classical Multilayer Perceptron structure [8] in order to operate on interval-valued input and output data. Thus, it allows to deal with variability (in interval form) in the data set.

Other perceptrons dealing, in some manner, with intervals have been proposed, for example, in [9] and [10]. The one proposed by Beheshti et al. [10] is the more similar to the iMLP, as it has inputs and outputs in interval form. However, in the Behesti perceptron the weights and biases are intervals, whereas, in the iMLP, they are crisp values. Consequently, the estimation of the optimal weights and biases in the Behesti perceptron is done by means of interval computational algorithms and is substantially more complex than the calibration of the iMLP. Thus, the iMLP will be considered to forecast ITS and will be described below.

6.1 The Interval Multilayer Perceptron

An iMLP with n inputs and m outputs is comprised of an input layer with n input buffer units, one or more hidden layers with a non-fixed number of nonlinear hidden units and one output layer with m linear or nonlinear output units. Henceforth, we will consider just one hidden layer with h hidden units and one output ($m = 1$). The operations in the iMLP follow the rules of interval arithmetic (see section 3).

Considering n interval-valued inputs $[X]_i = \langle X_{i,C}, X_{i,R} \rangle = [X_{i,C} - X_{i,R}, X_{i,C} + X_{i,R}]$, with $i = 1, .., n$, the output of the j-th hidden unit is the weighted linear combination of the n interval inputs and the bias. It is worth noting that

the weights of the proposed structure are crisp and not intervals. The linear combination results in a new interval given by:

$$[S]_j = w_{j0} + \sum_{i=1}^{n} w_{ji}[X]_i = \langle w_{j0} + \sum_{i=1}^{n} w_{ji}X_{i,C}, \sum_{i=1}^{n} |w_{ji}|X_{i,R}\rangle.$$

The activation of the j-th hidden unit is obtained by transforming the interval $[S]_j$ using a nonlinear activation function, more precisely, the tanh function. This function is monotonic, then the interval output is given by $f([A]) = [f(A_L), f(A_U)]$. Thus, the resulting interval can be calculated as:

$$\begin{aligned}
[A]_j &= \tanh([S]_j) = [\tanh(S_{j,C} - S_{j,R}), \tanh(S_{j,C} + S_{j,R})] = \\
&= \langle \frac{\tanh(S_{j,C} - S_{j,R}) + \tanh(S_{j,C} + S_{j,R})}{2}, \\
&\quad \frac{\tanh(S_{j,C} + S_{j,R}) - \tanh(S_{j,C} - S_{j,R})}{2} \rangle.
\end{aligned}$$

The output of the network, $[\hat{Z}]$, is obtained by transforming the activations of the hidden units using a second layer of processing units. In the case of a single output and a linear activation function, the estimated output interval is a linear combination of the activations of the hidden layer and the bias:

$$[\hat{Z}] = \sum_{j=1}^{h} \alpha_j [A]_j + \alpha_0 = \langle \sum_{j=1}^{h} \alpha_j A_{j,C} + \alpha_0, \sum_{j=1}^{h} |\alpha_j| A_{j,R} \rangle.$$

The iMLP can be used to approximate an interval-valued function. The iMLP crisp weights can be adjusted with a supervised learning procedure by minimizing an error function of the form:

$$E = \frac{1}{p} \sum_{t=1}^{p} d([Z]_t, [\hat{Z}]_t) + \lambda \Phi(\hat{f}),$$

where $d([Z]_t, [\hat{Z}]_t)$ is a measure of the discrepancy between the actual and the estimated output intervals for the t-th training sample with $t = 1, ..., p$; and $\lambda\Phi(\hat{f})$ is a regularization term [11] of the estimated function $\hat{f}([X]_i) : [X]_i \rightarrow [Z]$ with $i = 1, ..., n$. A weighted Euclidean distance function for a pair of intervals $[A]$ and $[B]$ can be used as discrepancy measure:

$$d([A], [B]) = \beta(A_C - B_C)^2 + (1 - \beta)(A_R - B_R)^2.$$

The parameter $\beta \in [0, 1]$ allows to assign more weight to the error in the centers or in the radii. This discrepancy function can be minimized applying a low-memory Quasi Newton method [12] with random initial weights. Second order methods require the calculation of the gradient of the cost function with respect to the adaptive weights ($w's$ and $\alpha's$). These derivatives can be calculated in an

effective way by applying a backpropagation procedure, similar to this proposed in [8] for the standard MLP. More details are given in [7].

Due to the capability of input-output mapping of the iMLP, it can be used for causal forecasting of ITS or for extrapolative ITS forecasting. In the second case, the functional relationship to be estimated by an iMLP can be written as $[X]_{t+1} = f([X]_t, [\hat{X}]_{t-1}, [X]_{t-l})$, where $[X]_t$ is the interval observed at time t.

7 Analysis of the forecasting performance

The original data set consists of records of the monthly mean temperature in 60 weather stations in China from January 1952 to December 1988 (i.e. 444 months). These stations make up a network with a relatively uniform spatial distribution and each one is representative of a particular climate region of China. Data can be obtained in the archive of the Computational and Information Systems Laboratory (http://dss.ucar.edu/datasets/ds578.5/data/). The 60 temperature time series has been aggregated leading to an ITS of 444 monthly periods, where each period represents the interval of the monthly mean temperature throughout China. Figure 1 shows that the ITS has a seasonal pattern with $s = 12$ that concerns not only the interval centers, but also the ranges; it is clear that summer months have less range than winter months.

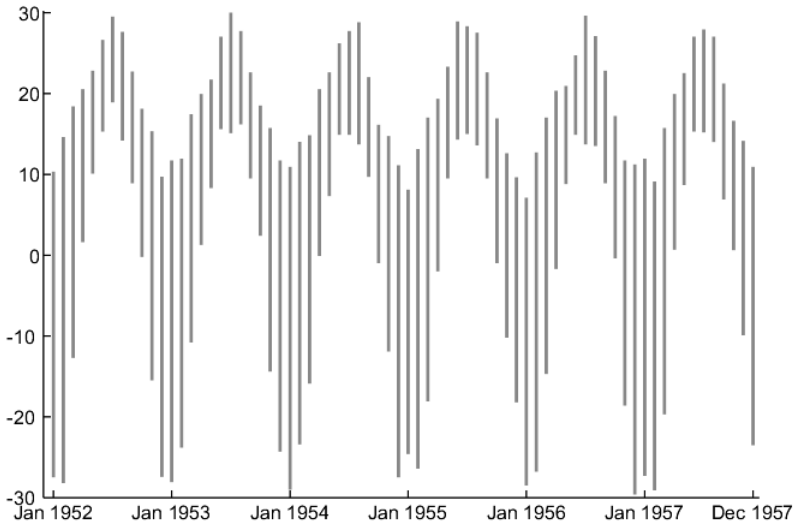

Fig. 1: ITS of the monthly temperature in China (1952-1957).

The training set consists of the first 296 periods, while the test set contains the last 148 periods. The ITS has been forecasted using different approaches:

1. naive model with seasonality: $[\hat{X}]_{t+1} = [X]_{t-s+1}$
2. IES with crisp seasonality: $\alpha = 1$, $\delta = 0.93$

3. IES with interval seasonality: $\alpha = 0.17$, $\delta = 0.19$ and $\xi = 0.03$
4. iMLP with 3 layers ($n = 6, h = 6, m = 1$) and using as inputs $[X]_{t-l+1}$, where $l = \{1, 2, 12, 13, 24, 25\}$,
5. modelling separately the minimum and maximum series:
   (a) exponential smoothing models with additive level and seasonality
      - minimum: $\alpha = 0.101, \delta = 0.1075$
      - maximum: $\alpha = 0.028, \delta = 0.16$
   (b) ARIMA models
      - minimum: ARIMA $(1,0,1)(2,1,1)_{12}$ without constant
      - maximum: ARIMA $(1,0,0)(0,1,1)_{12}$ with constant
6. modelling separately the centers and radii series with:
   (a) exponential smoothing models with additive level and seasonality
      - centers: $\alpha = 0.0865, \delta = 0.1274$
      - radii: $\alpha = 0.0602, \delta = 0.1254$
   (b) ARIMA models
      - centers: ARIMA $(1,0,1)(2,1,1)_{12}$ without constant
      - radii: ARIMA $(1,0,0)(0,1,1)_{12}$ with constant

Table 1 summarizes the forecasting performance of the considered approaches. The IEScS obtains a forecasting performance worst than the seasonal naive; this is due to the fact that the ITS seasonality affects both, interval range and interval center, and not only centers as the IEScS assumes. The iMLP outperforms the seasonal naive model and the IEScS, but it is less accurate than the rest of the methods. Modelling the univariate series with ARIMA models is the best method in this case. The IESiS obtains a good result, especially, if we consider that it only requires 3 parameters instead of the 8 needed by the ARIMA based approaches. The performance of the IESiS is quite similar to the performance of the models that forecast the univariate series with exponential smoothing methods, but our method is slightly simpler and deals with intervals as a whole.

8 Conclusions

ITS provide a way of modelling the range variation of an observed phenomenon through time. The proposal of methods to forecast and to analyze ITS is an interesting challenge. In this paper, an extension of exponential smoothing methods to ITS has been proposed and the iMLP has been applied to ITS forecasting. The forecasting performance of these methods is promising but it must be improved in the future. We believe that forecasting methods for ITS must deal with intervals as a whole. Therefore, more sophisticated ITS forecasting methods should be proposed. Teles and Brito [1] adapted ARMA models to ITS, a comparison of the accuracy of exponential smoothing and ARMA models in ITS must be done in the future.

model	training	test
1- seasonal naive	2.348	2.432
2- IEScS	3.675	3.45
3- IESiS	1.88	1.729
4- iMLP	2.08	2.07
5a- min-max (exp. smooth.)	1.856	1.703
5b- min-max (ARIMA)	1.508	1.554
6a- cen-rad (exp. smooth.)	1.841	1.669
6b- cen-rad (ARIMA)	1.553	1.539

Table 1: Forecasting performance in terms of the MDE_H.

References

[1] P. Teles and M. P. Brito. Modelling interval time series data. In *3rd IASC world conference on Computational Statistics & Data Analysis*, Limassol, Cyprus, 2005.

[2] J. Arroyo and C. Maté. Introducing interval time series: Accuracy measures. In *COMPSTAT 2006, Proceedings in Computational Statistics*, pages 1139–1146, Heidelberg. Physica-Verlag.

[3] H.-H. Bock and E. Diday, editors. *Analysis of Symbolic Data. Exploratory Methods for Extracting Statistical Information from Complex Data*. Studies in Classification, Data Analysis, and Knowledge Organization. Springer, Berlin Heidelberg, first edition, 2000.

[4] R.. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[5] E. S. Gardner. Exponential smoothing: The state of the art. part 2. *International Journal of Forecasting*, 22(4):637–666, 2006.

[6] G. Zhang, B. E. Patuwo, and M. Y. Hu. Forecasting with artificial neural networks: The state of art. *International Journal of Forecasting*, 14(1):35–62, 1998.

[7] A. Muñoz San Roque, C. Maté, J. Arroyo, and A. Sarabia. imlp: Applying multi-layer perceptrons to interval-valued data. Technical report (submitted to neural processing letters, july 2006), Instituto de Investigación Tecnológica, Universidad Pontificia Comillas, 2006.

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[9] H. Ishibuchi, H. Tanaka, and H. Okada. An architecture of neural networks with interval weights and its application to fuzzy regression analysis. *Fuzzy Sets and Systems*, 57:27–39, 1993.

[10] M. Beheshti, A. Berrached, A. de Korvin, C. Hu, and O. Sirisaengtaksin. On interval weighted three-layer neural networks. In *Annual Simulation Symposium*, pages 188–195. IEEE Computer Society, 1998.

[11] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

[12] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, Reading, MA, 1984.

An empirical comparison between two systems for automatic ARIMA modelling and forecasting

H. Njimi[1] and G. Mélard[2] *

1- ISRO CP 210 (bldg NO room 2.O.9.216)
Campus Plaine, Université Libre de Bruxelles
Bd du Triomphe
B-1050 Bruxelles (BELGIUM)
Email: hnjimi@ulb.ac.be
2- ECARES CP 114 (Solbosch bldg S room S.11.131)
Université Libre de Bruxelles
Avenue Franklin Roosevelt, 50
B-1050 Bruxelles (BELGIUM)
Email: gmelard@ulb.ac.be
Homepages: http://homepages.ulb.ac.be/~gmelard/

Abstract. The paper consists in an empirical comparison of two automatic procedures of time series forecasting: the expert system called TSE-AX and the automatic procedure of the TRAMO-SEATS software package. That comparison is based on the data of the M3-Competition, the latest of the M-Competitions launched by Spyros Makridakis and Michèle Hibon. TSE-AX was a competitor in the M3-Competition but an improved version has been used here. TRAMO-SEATS is a pair of programs initially aimed at seasonal decomposition of quarterly or monthly time series, using a signal extraction approach based on ARIMA modelling. We use only the automatic procedure within TRAMO. In this paper, we briefly describe the principle of each procedure before giving the results of that comparison. It is interesting that two procedures based on different strategies provide similar results, on the whole, and that TRAMO, which was not intended as a forecasting software package, appears as a very satisfactory forecasting solution.

Key words. Time series prediction, M3-Competition, Forecasting methods, TSE-AX, TRAMO-SEATS

1 Introduction

Forecast errors can have harmful consequences and imply, for example, surplus production capacity, under-capacity, out of stock items or unsold goods. If it is impossible to eliminate them completely, reliability of the forecasts can however be increased by applying good principles resulting from research and practice [1]. These should indicate which methods to rely on and specify the conditions for their optimal use.

To forecast data in economics and finance, several statistical and econometric methods are used such as regression models, multivariate analysis, decision theory or

* This paper has benefited from an IAP-network in Statistics grant, contract P5/24, Belgian Federal Office for Scientific, Technical and Cultural Affairs.

time series modelling. Among the later, we consider ARIMA processes made popular by Box and Jenkins who proposed a model building methodology composed of several stages [2]. ARIMA modelling is more difficult to use than other statistical forecasting techniques although, when implemented properly, it can be quite powerful and flexible. On the basis of Eurostat data bases, Fischer and Planas [3] have argued that the so-called airline model can be used to fit a large number of series. Several algorithms of automated ARIMA modelling were developed in order to make the method more applicable and also available to a greater number of users. Most of these algorithms were implemented using expert systems technology. These systems make it possible to program the knowledge of an expert and to reproduce the reasoning carried out by the system. One of these expert systems for building univariate time series models is TSE-AX. Described by Mélard and Pasteels [4], it is included in Time Series Expert 2.3. A slightly improved version [5] is used here for which Njimi et al. [6] gave an early presentation. One of the recent features is the possibility to handle series other than monthly and quarterly, as illustrated by Azrak et al. [7].
TSE-AX [4] was well ranked in the M3-Competition where the participating experts were asked to make forecasts beyond the available data [8, 9]. Note that the real data corresponding to these forecasts were not available to the participants before making their forecasts and were not, therefore, used in developing their forecasting model.
In this paper we present an empirical comparison between TSE-AX [5] and the automatic procedure of TSW, the Windows version of TRAMO-SEATS with some modifications and additions, developed by Caporello et al. [10] at the Banco de España.
Note that TRAMO-SEATS is a pair of programs initially aimed at seasonal decomposition of monthly or lower frequency time series, using a signal extraction approach based on ARIMA modelling. We use only the automatic procedure within TRAMO, which was not intended as a forecasting software package. That comparison is based on a subset of the series of the M3-Competition.
The contents of the paper are as follows. First, we give a description of TSE-AX. Second, we describe the automatic procedure of TSW [11]. And finally we give the principle of the comparison between those two automatic procedures and some results.

2 Description of TSE-AX

The objective of TSE-AX [5] is to build ARIMA models in an automated way, with and without an intervention analysis, but so that the user receives the intermediate and final results, and is informed of the quality of the final model. The system is adapted to several categories of users from beginners to experts. The later should use such a tool to save time, being qualified to assess the quality of the final model and possibly propose an alternative model. Briefly, TSE-AX covers everything from the specification stage to the forecasting stage, given that the latter is immediate when a final model has been found. The user can specify his or her model building preferences: perform an intervention analysis or not, choose a specification strategy, etc.
The modelling stage of TSE-AX consists of a succession of several phases. At the beginning, the user gives some information to the system like periodicity of the data

and the sample to be used. The automated procedure starts with the preliminary stage, where interventions are selected, transformations are performed, and differences, regular and/or seasonal, are chosen to make sure that the series becomes stationary. To select differences, the user can choose between options based on the non-parametric test of Kruskal and Wallis [12] and presence of autocorrelations or comparison of variances [4, 5]. Next follows the specification stage, where an ARIMA model is identified using one of these three strategies: 'expert' [13], 'autoregressive specification' [14] and 'mixed' [5] where a certain number of models are fitted and a choice among them is made. The remaining stages are the estimation stage, where the final model is fitted, the model checking stage, were the adequacy of that model is investigated, and the forecasting stage. In all modelling stages, the parameters are estimated by exact maximum likelihood.
There are more than twenty input commands that enable the user to customize the modelling strategy. They are concerned with the treatment of outliers by intervention analysis (several types are supported like additive outliers (AO) and level shift (LS)), the seasonal component, the Box-Cox transformation and difference operators. These commands are typically entered into a file and can act either on a single series or on a stream of series. Here, default values for all commands were used.

3 Description of TSW and its automatic procedure

TSW is a Windows interface that integrates the two programs TRAMO and SEATS. The software and its documentation are freely available at the address http://www.bde.es/. TRAMO, "Time series Regression with ARIMA noise, Missing values and Outliers" [15, 16] is a program for fitting and forecasting of regression models with possibly non-stationary ARIMA errors and missing values. The program interpolates these values, identifies and corrects for several types of outliers, not only additive outliers (AO) and level shift (LS), but also temporary change (TC) and innovation outliers (IO), and estimates special effects such as trading day and Easter effects and, in general, intervention-variable type effects. SEATS, "Signal Extraction in ARIMA Time Series", is a program for extracting unobserved components in time series with the purpose to produce a seasonally adjusted series. TRAMO and SEATS are structured so as to be used together but TRAMO can be used alone. TRAMO preadjusts the series, and SEATS decomposes the linearized series into its stochastic components. The complete final component is equal to the stochastic one, plus the deterministic effect associated with that component, that has been removed in the preadjustment by TRAMO (for example, an AO outlier will be added to the irregular component, a LS outlier will be added to the trend-cycle, and so on).
The programs, TRAMO and SEATS, are fundamentally aimed at monthly or lower frequency time series. Although structured to meet the needs of an expert analyst, they can be reliably used in an entirely automatic manner on very large sets of time series. The main applications are seasonal adjustment, trend-cycle estimation, construction of leading indicators, interpolation, detection and correction of outliers, estimation of special effects, and quality control of data. It should be insisted that TRAMO-SEATS is not aimed at forecasting.
The automatic procedure of TSW requires the prior decision of whether or not to test for the presence of calendar effects and, if so, which specification for the trading day

effect should be used. The program tests for the log/level specification, interpolates missing observations (if any), and performs automatic model identification and outlier detection, see [10]. Three types of outliers are considered: additive outliers, transitory changes and level shifts; the level of significance is set by the program and depends on the length of the series. The full model is estimated by exact maximum likelihood, and forecasts of the series up to a two-year horizon are computed.
Within SEATS, the model obtained by TRAMO is decomposed and optimal estimators and forecasts of the components are obtained, as well as their mean squared error. These components are the trend-cycle, and the seasonal, irregular and (perhaps) transitory components. If the model does not accept an admissible decomposition, it is replaced by a decomposable one.

4 Modelling methodology

Our analysis of the M3-Competition series is limited to yearly, quarterly and monthly series, i.e. 2829 series out of 3003. We recall that the 3003 series of the M3-Competition were selected on a quota basis to include various types of time series data (micro, industry, macro, etc.) and different time intervals between successive observations (yearly, quarterly, etc.), see [8]. Usually, yearly series are discarded because most of them are too short to be modelled by ARIMA models with existing technology but here we kept them all. Series with an unknown time interval between successive observations are excluded because both automatic procedures, of TSE-AX and TSW, require that information.
For the treatment by TSE-AX, we have accepted the treatment of outliers by intervention analysis to avoid extreme values that would badly influence the various steps of the analysis: specification, estimation, test for adequacy, and forecasting. We also used the 'Mixed strategy' in the step of specification because that strategy is the most complete one [6]. For the treatment by TSW, we choose the option for automatic model identification, as explained above. Even if the computational strategies of computation of TSE-AX and TSW are not the same, the choice of that option is justified by the fact that this is the most general without calendar effects. This means that these two automatic procedures include a treatment of outliers and use exact maximum likelihood for estimation of the final model, and no pretest is made for the presence of trading day, leap year and Easter effects.
All fits are done in TSE-AX by exact maximum likelihood estimation whereas the Hannan-Rissanen estimation method is used in TSW for all fits except the final one which is performed by exact maximum likelihood estimation. Using the mixed strategy, TSE-AX fits at most 32 models. The number of models fitted by TSW is much larger but computations are faster.

5 Results of the comparison

As in the M-Competition, we selected a forecasting horizon (h) of maximum six years for yearly series, eight quarters for quarterly series, and eighteen months for monthly series. The symmetric mean absolute percentage error criterion (sMAPE) is used to analyze the performance of the two procedures. The sMAPE is an average across all forecasts made for a given horizon i :

$$\frac{1}{N}\sum_{i=1}^{N}\frac{|X_i - F_i|}{(X_i + F_i)/2}\times 100$$

where X_i is the real value, F_i is the forecast and N is the number of predicted time series.

For each type of series, for each procedure and for each horizon, the sMAPE is calculated. The results are summarized in Table 1 for yearly series, Table 2 for quarterly series and Table 3 for monthly series (see Appendix). Tables 4-6 list the differences in terms of forecasting performance of TSW with respect to TSE-AX. They show sMAPE(TSE-AX) – sMAPE(TSW) using the results in Tables 1-3: a positive sign means that the accuracy of TSW is better than that of TSE-AX. Tables 7-9 give the resulting *p*-values from the means paired test, using the average results of the comparison between TSE-AX and TSW based on the sMAPE criterion for, respectively, yearly, quarterly and monthly data.

5.1 Yearly data

The results of the yearly series are shown in Tables 1, 4 and 7. For these series, the results suggest that TSW did worse than TSE-AX for macro series and did better in finance and other series.

Note that the differences in the forecasting performance (as far as the sMAPE is concerned) between the two automatic procedures are small and the maximum of these differences across types and horizons is 1.91%. Note also that the results of these two methods appear better than those of the other competitors in the M3-Competition, but, of course, our analysis is ex post.

5.2 Quarterly data

The results of the quarterly series are shown in Tables 2, 5 and 8. For these series, the results suggest that TSW did worse than TSE-AX for finance and demographic series and did better in micro, macro and industrial series.

The differences in the forecasting performance between the two automatic procedures are small and the maximum of these differences across types and horizons is 2.52%.

5.3 Monthly data

The results of the monthly series are shown in Tables 3, 6 and 9. For these series, TSW did worse for finance, demographic and other series, and did better in micro, macro and industrial series. The differences in terms of forecasting performance are also small. Those differences did not exceed 1.4%.

5.4 Statistical significance of the results

Instead of just comparing the sMAPE given by the two automatic procedures, we have performed statistical tests to compare their performance, using more precisely the paired *t*-test for every type of series and every horizon, except for the 'other' category for yearly series. Indeed the latter contains only 11 series, therefore the *p*-values are provided by the Wilcoxon signed rank test instead of the paired *t*-test. The *p*-value of each case (per type of series and horizon) is displayed in Tables 7-9. Small

values imply that the null hypothesis of equal performance is rejected and thus the accuracies of the two automatic procedures are significantly different.
All values are greater than 10% except in microeconomic series and monthly industrial series. More precisely, the *p*-values are less than 5% for horizon 5 in yearly macro data, horizons 1 and 4 in quarterly micro data, for horizon 5 in monthly micro data and for horizons 3 and 4 in monthly industrial data. The winner is indicated in bold in Tables 1-3.
In all remainder cases, the *p*-values are not significant. Given the large number of tests performed, this suggests that the difference in the forecasting performance of the two automatic procedures is small. A more formal comparison using a stepwise multiple testing procedure by Romano and Wolf [17] is given in [18].

6 Closing comments and conclusion

In Section 3, we have provided a rough description of TSW and its module TRAMO in particular. TRAMO is aimed at building an ARIMA model for the signal extraction procedure within SEATS in order to obtain a seasonal decomposition of a series. Although TRAMO is not considered as a forecasting software package, it seemed interesting to investigate that complex procedure and its forecasting performance.
Having performed a comparison between the expert system TSE-AX and the automatic procedure of TSW on the yearly, quarterly and monthly series of the M3-Competition, it can be concluded that there is no statistically significant difference in the forecasting performance except perhaps for microeconomic data where TSW performs slightly better.
Even if the computational strategies of TSE-AX and TSW are not the same, they produce comparable forecasts on the whole. Therefore, given that TSE-AX was well assessed in the competition, that implies that TSW can produce good forecasts.
Again, this comparison illustrates the fact that the "use of statistically sophisticated or complex methods does not necessarily produce consistently more accurate forecasts" [19].

References

[1] Armstrong J. S., Editor (2001). Principles of Forecasting, A Handbook for Researchers and Practitioners, Springer, New York.

[2] Box G. E. P., Jenkins G. M., Reinsel G. C. (1994). *Time Series Analysis, Forecasting and Control*, 3rd ed., Prentice-Hall Press, Englewood Cliffs, NJ.

[3] Fischer B., Planas C. (2000). Large scale fitting of regression models with ARIMA errors. *Journal of Official Statistics* 16, 173-184.

[4] Mélard G., Pasteels J.-M. (2000). Automatic ARIMA modeling including interventions, using time series expert software, *International Journal of Forecasting* 16, pp. 497-508.

[5] Njimi H., Mélard G., Pasteels J.-M. (2006). User's Manual of Time Series Expert (TSE version 2.4) – TSE-AX: An expert system for ARIMA models. Institut de Statistique et de Recherche Opérationnelle, Université Libre de Bruxelles, Bruxelles, to be published.

[6] Njimi H., Mélard G., Pasteels J.-M. (2003). Modélisation SARIMA assistée, *Actes des XXVèmes Journées de Statistique, Lyon, France, 13-17 mai 2003*, Société Française de Statistique, Tome 2, pp. 731-734.

[7] Azrak R., Mélard G., Njimi H. (2004). Forecasting in the analysis of mobile telecommunication data - Correction for outliers and replacement of missing observations, *Journal Marocain d'Automatique, d'Informatique et de Traitement du Signal,* numéro spécial CoPSTIC'03, Première Conférence Plénière du Pôle des Compétences en Sciences et Technologies de l'Information et de la Communication, novembre 2004, 1-14.

[8] Makridakis S., Hibon M. (2000). The M3-Competition: Results, conclusions and implications. *International Journal of Forecasting* 16, 451–476.

[9] Ord K., Hibon M., Makridakis S. (2000). The M3-Competition. *International Journal of Forecasting* 16, 433–436.

[10] Caporello G., Maravall A., Fernando S. (2001). Program TSW Reference Manual, Banco de España, Madrid.

[11] Gómez V., Maravall A. (2001). Automatic Modelling Methods for Univariate Series, in Peña D., Tiao G. C., Tsay R. S. (eds.) *A Course in Advanced Time Series Analysis*, Wiley, New York, pp. 202-246.

[12] Kruksal W. H., Wallis W. A. (1952). Use of ranks in the one-criterion variance analysis. *Journal of the American Statistical Association* 67, 401–412.

[13] Mélard G., Pasteels J-M. (1998). User's Manual of Time Series Expert (TSE version 2.3). Institut de Statistique et de Recherche Opérationnelle, Université Libre de Bruxelles, Bruxelles.

[14] Mélard G. (1990). *Méthodes de prévision à court terme*. Editions de l'Université de Bruxelles, Bruxelles, and Editions Ellipses, Paris.

[15] Gómez V., Maravall A. (1994), *Estimation, prediction, and interpolation for nonstationary series with the Kalman filter*, Journal of the American Statistical Association 89, 611-624.

[16] Gómez V., Maravall A. (1996). Programs TRAMO and SEATS: Instructions for the User (with some updates). Working Paper 9628, Research Department, Banco de España.

[17] Romano J., Wolf M. (2005). Control of generalized error rates in multiple testing. Technical Report 2005-12. Department of Statistics, Stanford University.

[18] Njimi H., Mélard G. (2006). Forecasting competitions: a suggested procedure for multiple testing. Communication to International Symposium on Forecasting, ISF2006, Santander (Spain), June 11-14, 2006.

[19] Flores B. E., Pearce S. L. (2000). The use of an expert system in the M3-Competition. *International Journal of Forecasting* 16, 485–496.

Appendix

Types of series	Methods	Horizon					
		1	2	3	4	5	6
Micro	TSE-AX	12,34	16,19	20,89	23,62	26,29	28,77
	TSW	11,45	15,66	21,07	23,59	26,74	28,75
Industrial	TSE-AX	9,60	11,81	14,05	16,69	18,59	20,45
	TSW	10,79	12,68	14,68	15,72	17,40	19,31
Macro	TSE-AX	2,40	3,68	4,92	5,82	6,51	7,09
	TSW	2,57	3,93	5,33	6,32	7,12	7,79
Finance	TSE-AX	17,16	20,59	22,06	24,37	27,00	29,09
	TSW	17,02	20,11	21,93	23,91	25,35	27,18
Demographic	TSE-AX	4,87	6,08	7,46	9,16	10,75	12,13
	TSW	5,36	6,33	7,55	8,66	10,07	11,28
Other	TSE-AX	16,57	19,17	21,80	22,71	22,02	22,47
	TSW	15,48	17,98	21,17	22,26	21,75	22,30

Table 1: sMAPE for yearly data. Bold numbers are those significantly best at 5% in the sense of Section 5. 4.

Types of series	Methods	Horizon						
		1	2	3	4	5	6	8
Micro	TSE-AX	10,16	10,79	11,63	12,83	13,32	13,98	15,08
	TSW	7,64	8,64	9,44	10,56	11,08	11,85	13,25
Industrial	TSE-AX	6,63	7,74	8,10	8,29	8,85	9,91	11,22
	TSW	5,76	6,34	6,92	7,38	7,87	8,53	9,79
Macro	TSE-AX	2,63	3,01	3,52	3,96	4,45	4,87	5,89
	TSW	2,48	3,00	3,53	3,93	4,36	4,76	5,71
Finance	TSE-AX	5,08	8,79	10,14	11,81	13,41	15,08	16,84
	TSW	5,79	9,05	10,87	12,39	13,89	15,08	17,12
Demographic	TSE-AX	5,09	7,21	8,45	9,81	11,40	12,78	15,61
	TSW	6,86	8,84	10,02	11,55	13,27	14,52	17,25
Other	TSE-AX	No series						
	TSW							

Table 2: sMAPE for quarterly data. Bold numbers are those significantly best at 5% in the sense of Section 5. 4.

Types of series	Methods	Horizon									
		1	2	3	4	5	6	8	12	15	18
Micro	TSW	24,60	23,81	24,24	24,41	23,86	23,68	23,51	23,81	25,31	26,48
	TSE-AX	25,19	24,56	24,87	25,51	25,28	24,74	24,40	24,36	25,31	26,50
Industrial	TSW	7,14	7,44	7,72	8,18	8,61	9,23	10,25	11,18	11,95	12,69
	TSE-AX	7,61	7,95	8,39	8,89	9,26	9,65	10,61	11,48	12,21	12,97
Macro	TSW	3,03	3,52	3,78	3,77	4,10	4,38	4,82	5,34	5,92	6,69
	TSE-AX	3,09	3,52	3,58	4,01	4,31	4,53	4,96	5,60	6,13	6,83
Finance	TSW	5,90	6,88	7,07	7,61	8,49	8,88	9,79	10,92	11,75	12,89
	TSE-AX	5,76	6,59	6,59	7,27	8,23	8,69	9,60	10,37	11,30	12,25
Demographic	TSW	2,55	2,82	3,30	4,13	4,91	5,35	6,10	6,98	7,56	8,44
	TSE-AX	2,62	2,98	3,33	4,09	4,80	5,16	5,71	6,52	7,01	7,78
Other	TSW	5,84	7,04	6,08	6,34	6,54	9,07	9,63	9,33	9,29	10,23
	TSE-AX	5,94	6,75	5,78	6,10	6,37	8,69	9,92	9,15	8,97	9,70

Table 3: sMAPE for monthly data. Bold numbers are those significantly best at 5% in the sense of Section 5. 4.

Types of series	Horizon					
	1	2	3	4	5	6
Micro	0,89	0,53	-0,18	0,04	-0,45	0,02
Industrial	-1,19	-0,87	-0,63	0,97	1,18	1,14
Macro	-0,17	-0,25	-0,41	-0,51	-0,62	-0,70
Finance	0,14	0,48	0,13	0,47	1,65	1,91
Demographic	-0,50	-0,25	-0,09	0,50	0,68	0,86
Other	1,10	1,19	0,63	0,45	0,27	0,17

Table 4. sMAPE(TSE-AX) – sMAPE(TSW): yearly data. Bold numbers are those significantly best at 5% in the sense of Section 5. 4.

Types of series	Horizon						
	1	2	3	4	5	6	8
Micro	2,52	2,14	2,19	2,27	2,24	2,13	1,83
Industrial	0,86	1,41	1,18	0,91	0,99	1,38	1,43
Macro	0,15	0,01	-0,01	0,03	0,10	0,10	0,18
Finance	-0,71	-0,26	-0,73	-0,58	-0,48	0,00	-0,28
Demographic	-1,77	-1,62	-1,57	-1,74	-1,87	-1,74	-1,64
Other	No series						

Table 5. sMAPE(TSE-AX) – sMAPE(TSW): quarterly data. Bold numbers are those significantly best at 5% in the sense of Section 5. 4.

Types of series	Horizon									
	1	2	3	4	5	6	8	12	15	18
Micro	0,58	0,74	0,63	1,10	1,42	1,06	0,89	0,55	0,00	0,02
Industrial	0,47	0,51	0,67	0,72	0,65	0,42	0,36	0,31	0,26	0,29
Macro	0,06	0,01	-0,19	0,24	0,21	0,15	0,14	0,26	0,21	0,14
Finance	-0,13	-0,29	-0,47	-0,34	-0,26	-0,19	-0,20	-0,54	-0,45	-0,65
Demographic	0,07	0,15	0,03	-0,05	-0,11	-0,19	-0,39	-0,46	-0,56	-0,66
Other	0,10	-0,28	-0,31	-0,24	-0,17	-0,38	0,29	-0,19	-0,32	-0,53

Table 6. sMAPE(TSE-AX) – sMAPE(TSW): monthly data. Bold numbers are those significantly best at 5% in the sense of Section 5. 4.

Types of series	Horizon					
	1	2	3	4	5	6
Micro	0,41	0,64	0,88	0,98	0,77	0,99
Industrial	0,14	0,29	0,46	0,32	0,25	0,33
Macro	0,37	0,25	0,10	0,06	0,04	0,05
Finance	0,93	0,78	0,95	0,82	0,48	0,44
Demographic	0,12	0,47	0,83	0,43	0,34	0,29
Other	0,08	0,32	0,41	0,41	0,72	0,61

Table 7: Resulting *p*-values from means paired test. Results of the comparison between TSE-AX and TSW based on the sMAPE criterion: yearly data

Types of series	Horizon						
	1	2	3	4	5	6	8
Micro	0,02	0,06	0,06	0,04	0,05	0,06	0,11
Industrial	0,39	0,16	0,22	0,34	0,34	0,22	0,21
Macro	0,26	0,91	0,94	0,82	0,56	0,59	0,43
Finance	0,23	0,69	0,13	0,15	0,26	0,99	0,69
Demographic	0,15	0,21	0,23	0,21	0,28	0,35	0,51
Other	No series						

Table 8: Resulting *p*-values from means paired test. Results of the comparison between TSE-AX and TSW based on the sMAPE criterion: quarterly data

Types of series	Horizon									
	1	2	3	4	5	6	8	12	15	18
Micro	0,48	0,26	0,31	0,08	0,02	0,05	0,09	0,33	0,99	0,98
Industrial	0,18	0,12	0,04	0,04	0,05	0,20	0,31	0,73	0,48	0,49
Macro	0,84	0,98	0,64	0,41	0,53	0,69	0,72	0,49	0,58	0,72
Finance	0,77	0,52	0,15	0,30	0,45	0,60	0,64	0,31	0,46	0,32
Demographic	0,74	0,40	0,85	0,80	0,55	0,30	0,10	0,19	0,15	0,13
Other	0,69	0,21	0,23	0,41	0,57	0,28	0,64	0,75	0,63	0,54

Table 9: Resulting *p*-values from means paired test. Results of the comparison between TSE-AX and TSW based on the sMAPE criterion: monthly data

Evaluating prediction models by parametric bootstrapping

Robert M. Kunst

University of Vienna - Dept of Economics
BWZ, Bruenner Strasse 72, 1210 Wien (Vienna) - Austria

Abstract. The traditional evaluation of competing prediction models relies on loss criteria for sample portions. The selected best prediction model class may not match the data-generating class.

Assuming one of the models as correct, simulating trajectories of pseudo-data (parametric bootstrapping) and forecasting them using each competing model class yields additional evidence. Comparisons of the original and the bootstrapped evaluation indicate whether a class describes the data-generating process (DGP) well. Of special interest are cases where the best prediction model class and the bootstrap DGP differ.

We apply the procedure to macroeconomic data and several time-series model specifications.

1 Introduction

Typically, forecasting relies on one of two basic concepts.

The first concept dominates the statistical literature, which often adds the task of prediction as a third main aspect of statistical work to the classical tasks of hypothesis testing and estimation. According to this view, a finite sequence of observations on a scalar or vector variable X is interpreted as a partial realization $X_{s+1}^{s+n}(\omega)$ of an unknown random process X_t—the *data-generating process* or DGP. It is used to obtain information on the distribution of a future sequence of the same random process, for example X_{r+1}^{r+m} with $r \geq s+n$. This formidable task can be made more modest by restricting attention to time-homogeneous processes, to specific classes of distributions, and to specific characteristics of the (conditional) distribution, such as moments or intervals with given probability and minimum length. In any case, an aspect of this view is that a potential knowledge of the DGP would solve the forecaster's problems.

The second concept is the naive practitioner's task. Given a sequence of observations X_{s+1}^{s+m}, on a scalar or a vector variable X, we wish to generate a sequence of *values* that is, in some sense, close to the not yet observed sequence X_{r+1}^{r+m}. Viewing the data as realizations of a stochastic process may help in determining a strategy for constructing an approximating sequence $\hat{X}_{r+1}^{r+m}$ but it can only be a means to an end. Mechanistic forecasting procedures, such as exponential smoothing, may deserve consideration, and knowing the DGP, even if such a one exists, does not necessarily solve the forecaster's problem, as she is interested in the realizations, not in the generating law.

An example may serve to highlight the discrepancies among concepts. Consider a fair die that is cast repeatedly. Each of a finite set of players is asked to bet a fixed amount of money and to predict the next face. If the value is predicted correctly, each correct forecaster receives an equal share of the total sum. If no one predicts correctly, the closest value wins and is also shared. In the statistical concept, the problem is solved by finding the correct uniform distribution of the experiment, which is conditional and also unconditional, as the throws are independent. The optimal point forecast would be 3.5. This information is of limited value to a better who is required to tell an integer number. In the naive forecaster's concept, the subtleties of the game play a role in designing a sequence of forecasts. Alternating values of three and four may optimize winning odds. If there are few players and the rules are modified, such that the bank receives all the money if more than one predict correctly, it may be preferable to alternate sequences between two and five or to randomize.

The dice example may serve as a good metaphor for the actual task of an economic forecaster who should predict next year's real economic growth rate. The correct conditional distribution with a mean of 2.2%, say, and a standard deviation of 2 percentage points, say, may fail to convince the media. The official forecast may be affected by the chance of a dramatic success—in case of a risk-loving institution—or by the prediction of the main competitors' forecasts—in case of a risk-averse institution. Even keeping the forecast close to a previous forecast may be beneficial on the forecasting market. In summary, public interest concentrates on future realized values, not on distributions, and the forecasts are assessed by complex loss functions and game-type interaction.

In this paper, we will refrain from these practical subtleties of the second approach and restrict focus on simple loss functions, such as the quadratic and the absolute loss. However, we wish to keep an eye on the shortcomings of the theory-based approach, particularly on the fact that the embedding of the data in a stochastic process is an auxiliary device rather than an obvious property of the observations.

The remainder of this paper contains two theory sections and an application section. Section 2 introduces parametric bootstrap as a tool that logically follows from considering ever more stochastic elements and more refinement in evaluating forecasts. Section 3 assesses the value of parametric bootstrap evaluations against the backdrop of model selection. Section 4 reviews the method in an application to national accounts aggregates. Section 5 concludes.

2 Basic concepts of forecast evaluation

It is easy to distinguish four concepts in empirical applications according to the technical scheme of the evaluations. For exposition, we base all evaluations on squared loss.

NN (non-stochastic data in the basic data and in the forecasts) evaluations

utilize statistics such as:

$$m^{-1}\sum_{j=1}^{m}\left(\hat{x}_{r+j}-x_{r+j}\right)^2, \tag{1}$$

where $\hat{x}_{r+j}$ may be 'rolling' one-step out-of-sample forecasts, for example. The statistic depends on the observations only. It is advisable to view it as a descriptive rather than a stochastic measure.

NS (non-stochastic forecasts but stochastic assumptions for the future data) evaluations rely on statistics such as:

$$\bar{R}^{-1}m^{-1}\sum_{R=1}^{\bar{R}}\sum_{j=1}^{m}\left(\hat{x}_{r+j}-x_{r+j}\left(\omega_R\right)\right)^2, \tag{2}$$

where the 'future' is drawn from a random processor. Typically, all $\bar{R}$ futures coincide with regard to the starting values x_t, $t \leq s+n$, or, by generalization, for all x_t, $t \leq s+n+j$ in rolling experiments for one-step predictions. The ensuing NS statistic is a random variable.

SN (stochastic forecasts but fixed data) evaluations are based on statistics such as:

$$\bar{S}^{-1}m^{-1}\sum_{S=1}^{\bar{S}}\sum_{j=1}^{m}\left(\hat{x}_{r+j}\left(\omega_S\right)-x_{r+j}\right)^2, \tag{3}$$

or, with a slight shift of emphasis, on:

$$m^{-1}\sum_{j=1}^{m}\left(\bar{S}^{-1}\sum_{S=1}^{\bar{S}}\hat{x}_{r+j}\left(\omega_S\right)-x_{r+j}\right)^2. \tag{4}$$

In the latter version, the average of the $\bar{S}$ stochastic replications serves as a mean predictor for x_{r+j}, while in the former version the distance of forecasting trajectories to future observations is highlighted. The latter version is more in line with traditional stochastic forecasting for nonlinear time-series models.

SS (all ingredients stochastic) evaluations are based on statistics such as:

$$\bar{R}^{-1}m^{-1}\sum_{R=1}^{\bar{R}}\sum_{j=1}^{m}\left(\bar{S}^{-1}\sum_{S=1}^{\bar{S}}\hat{x}_{r+j}\left(\omega_S\right)-x_{r+j}\left(\omega_R\right)\right)^2. \tag{5}$$

There are two principal strategies for drawing $x_{r+j}\left(\omega_R\right)$, that is, either by bootstrapping or by Monte Carlo based on sample estimates for specific model classes. The latter strategy is also called *parametric bootstrapping* in the following.

Each of the concept in turn answers different questions, such as:

1. Which forecasting strategy yields the best results for a given data set and variable?

2. Which forecasting strategy yields the best results for stochastic process-es that are similar to a given data set and variable—or: that are given by a specific Monte Carlo design?

3. Which (generally nonlinear) forecasting model yields the best results for a given data set and variable?

4. Which (generally nonlinear) forecasting model yields the best results for stochastic processes that are similar to a given data set and variable?

It may be tempting to conclude that the stochastic concepts NS and SS are more powerful than assessing prediction against the background of a data set only—the naive forecaster's view of section 1. The stochastic-data concepts, however, presuppose a certain degree of knowledge on the true DGP, which may also be viewed as a drawback. The pure data-driven evaluations SN and NN successfully isolate the task of prediction from the usual time-series analytic task of identifying the DGP. Presumably, their apparent lack of generality can be and should be overcome by considering parallel data sets, for example economic variables from various countries.

3 True and approximate processes

Figure 1 depicts the prototypical situation of empirical analysis. The plane represents the space of all possible models, not necessarily parametric with a two-dimensional parameter. Distance between points describes predictive accuracy, with longer distances corresponding to decreasing accuracy. This distance is not symmetric, as a model X may be better in predicting a true process Y than the other way around. Therefore, the representation is for illustration only, and distances of interest are marked by arrows heading to the process being predicted.

The point X represents the unknown true DGP, which is approximated within the model classes $M0$ and $M1$. In this diagram, $M0$ is a subset of $M1$, such that $M0$ can be viewed as a restricted version of $M1$. For example, $M1$ may be ARMA processes and $M0$ may be autoregressive processes. $X0$ and $X1$ represent pseudo-true parameter values within the classes. Typically, these pseudo-true values constitute optimal models for prediction. The points $X0$ and $X1$ are unknown, however, and are approximated by ML or other consistent estimates $Y0$ and $Y1$. The infeasible optimal predictors $X0$ and $X1$ are connected with the DGP at X by dotted arrows.

There is no guarantee that ML estimates optimize finite-sample predictive properties, such that points closer to X may be achievable by feasible methods. It may be, for example, that grid-search minimization of prediction loss over a test sample will result in such estimates. Even when ML estimates do optimize predictive accuracy—represented by Euclidian distance in the graph—within a class of estimators with reasonable properties, the ML estimate $Y1$ in $M1$ may be a worse approximation to X than $Y0$ in $M0$. As $n \to \infty$, $Y1$ and $Y0$ will

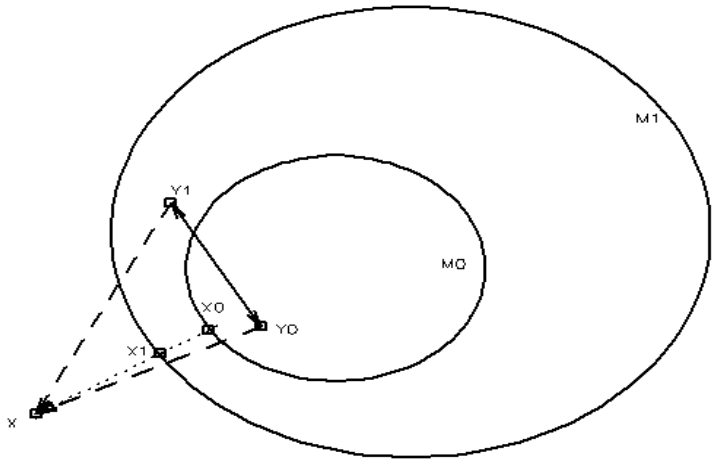

Fig. 1: Metaphorical representation of the forecaster's problem.

converge to the pseudo-true values under certain conditions (see, for example, White, 1994).

A less plausible backdrop for theoretical analysis would be the case where X actually is contained in $M1$. Even then, the outlined problem may persist in finite samples of empirical relevance.

Traditional SN methods are tools for determining the distance between X and the points $Y0$ and $Y1$. Unfortunately, the information on X is restricted by the sample size, which is expressed by dashing the arrows in Figure 1. Bootstrap SS methods measure the distance of $Y1$ and its approximation within $M0$, or of $Y0$ within $M1$. Due to sampling variation, even $Y1$ is not estimated by $Y1$ within $M1$ but via a feasible approximation. This approximation can be so poor that it is outperformed by a member of $M0$.

This is what appears to be happening in some experiments to be reported in the following. The evidence points to poor reliability of approximations of X within $M1$ and suggests using $M0$ instead. It is worth remarking that this ranking is typically affected by the position of X. A different X that is close to $M1$ but distant from the restricted set $M0$ can, of course, lead to its $M1$ approximation outperforming the $M0$ approximation. Within the metaphor of Figure 1, one may draw such an X in the northeast of the diagram, slightly outside the $M1$ boundary.

In this sense, SS bootstrapping is a convenient device for measuring the (asymmetric) distance of $Y1$ and $Y0$, in order to improve on the exactness of measuring the distance of X and its approximations, which is bounded by the available data of length n. In contrast, the (asymmetric) distance of $Y1$ and $Y0$

can be measured to an arbitrary degree of precision. This is expressed by the solid double arrow in Figure 1.

4 Empirical application: investment subaggregates

Jumah and Kunst (forthcoming) consider a system that consists of logarithmic transformations of three variables: gross domestic product (GDP) and two components of gross fixed capital formation and. They evaluate forecasts based on five competing time-series models and on UK data. We focus on four of those five models.

The first model is the simplest and is used as a backdrop. The three variables are differenced, and a vector autoregression is estimated for the differenced variables. The second model refines the first one by imposing a restriction on the deterministic constants such that mean growth rates of all three variables are identical.

The third model assumes co-integration between the individual investment components and GDP. Thus, it reflects the equilibrium concept that shares of the components in output remain constant in the long run. The observation that construction investment has shown a continuous decline over the recent decades, while equipment investment has increased its share remarkably, puts severe doubts on this hypothesis. Like the first two models, also the third one is linear. It is a standard error-correction VAR with two co-integrating vectors.

The fourth model assumes that the total share of investment in GDP remains constant while the two components may shift around. In order to impose such an equilibrium condition, Jumah and Kunst (forthcoming) utilize the nonlinear co-integration models that were introduced by Escribano and Mira (2002) and Escribano (2004).

While Jumah and Kunst (forthcoming) report that the nonlinear co-integration model clearly dominates the SN evaluation, Figures 2–5 show the results of an SS experiment for these four models and the UK data set. The criterion function was formed from the trace of an estimated $\mathrm{E}\hat{u}_t(h)\,\hat{u}_t(h)'$ matrix, where $\hat{u}_t(h)$ denotes the h–step prediction error. The scales of the three variables are sufficiently similar, such that the simple trace weighting hardly entails any distortion.

The data and also the bootstrapped pseudo-data have a sample size of 164 observations. 50 rolling out-of-sample forecasts are evaluated for horizons of 1 to 40. Original data are quarterly, thus the longest horizon of 40 quarters correspond to ten years. 100 replications are used for the bootstrap, and 100 replications are used for stochastic prediction and averaging. Pseudo-data were constructed according to estimates from the full original observations and starting conditions that were obtained from an early part of the data.

The MSE was evaluated on the basis of three ratios: total investment to GDP and each component to GDP. We note that the total investment quota is stationary in all models excepting the VAR in differences, while the component ratios are non-stationary in the non-linear co-integration model.

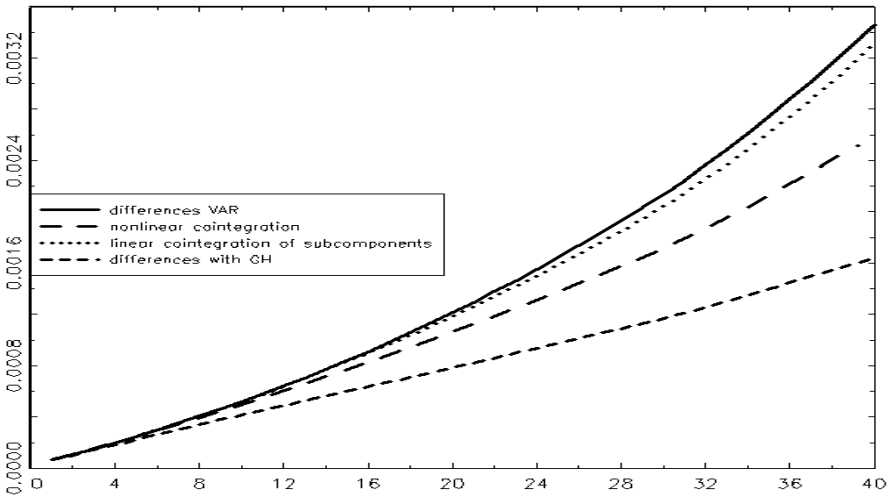

Fig. 2: Trace of the MSE matrix depending on prediction horizon. Generating model is linear VAR in differences.

Figure 2 relies on a parametric bootstrap of the simple VAR model in first differences without any additional constraints. Contrary to expectations, forecasts based on estimated structures of the generating class yield the largest prediction errors at all horizons. If the so-called growth homogeneity (GH) restriction is imposed in estimating the VAR, prediction errors are optimized. This outcome indicates that the data satisfy the GH constraint, such that even the unconstrained bootstrapped processes can be predicted efficiently by imposing GH. The linear co-integration model, which assumes stationary subcomponent quotas, performs almost as poorly as the unrestricted VAR in differences. The extrapolated bootstrapped trajectories do not satisfy the stationarity condition. The nonlinear model, which imposes co-integration on the total quota only, ranks in between the other methods. Freely bootstrapped VAR processes in differences do not satisfy any long-run restrictions, therefore the result is not entirely at odds with intuition.

Figure 3 shows that the linear VAR with GH wins its own experiment with flying colors. This graph is extremely similar to Figure 2, which indicates that growth homogeneity definitely holds within the observed data. It pays to impose this restriction in forecasting, as trajectories without the restriction tend to move apart. The nonlinear co-integration model also restrains deterministic growth effects, which implies an acceptable forecasting performance, while the unrestricted version of the DGP fails. The poor performance of the linear co-integration model is remarkable and confirms statistical evidence on the non-stationarity of subcomponent ratios in the data and thus also in the bootstrapped

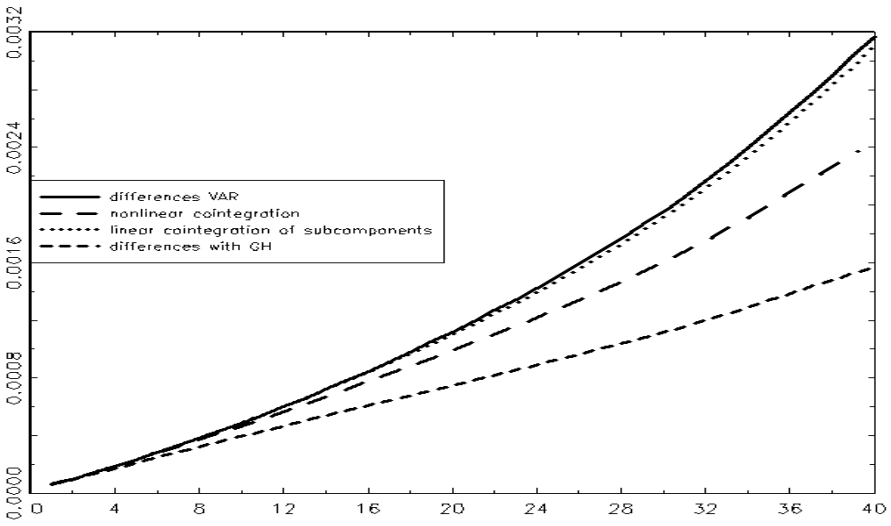

Fig. 3: Trace of the MSE matrix depending on prediction horizon. Generating model is linear VAR model in differences with growth homogeneity.

VAR structure.

Figure 4 shows that also the nonlinear co-integration model wins its own contest. The bootstrapped trajectories have time-constant total investment quotas but time-changing subcomponent quotas. The linear VAR with GH obeys neither of these conditions but the GH constraints guarantee that all quotas do not deviate too wildly from their in-sample values. It therefore outperforms the unconstrained linear model in differences. It also outperforms the linear co-integration model. This indicates that subcomponent ratios vary sufficiently, and not only in the data but also in bootstrapped trajectories that pick up some of the data properties, to violate the co-integrating condition. The criterion values are much smaller than in the other experiments, which confirms that the nonlinear model attains the closest fit to the data.

Figure 5 relies on simulated trajectories with time-constant subcomponent quotas. Nevertheless, the prediction performance of this model class itself is not satisfactory. The potentially misspecified model class—tentatively assumed as the correct specification—has small loading constants for the invalid co-integration vectors. Estimating these parameters constants results in large sampling variation, such that the co-integrating conditions do not help in prediction. Models that fail to impose co-integration, even though it is valid in the bootstrapped trajectories, dominate beyond $h = 30$.

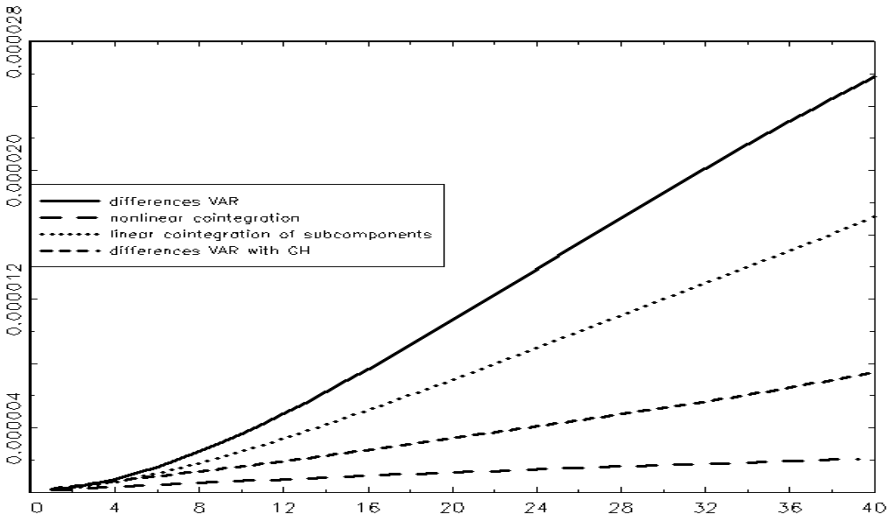

Fig. 4: Trace of the MSE matrix depending on prediction horizon. Generating model is nonlinear co-integration model.

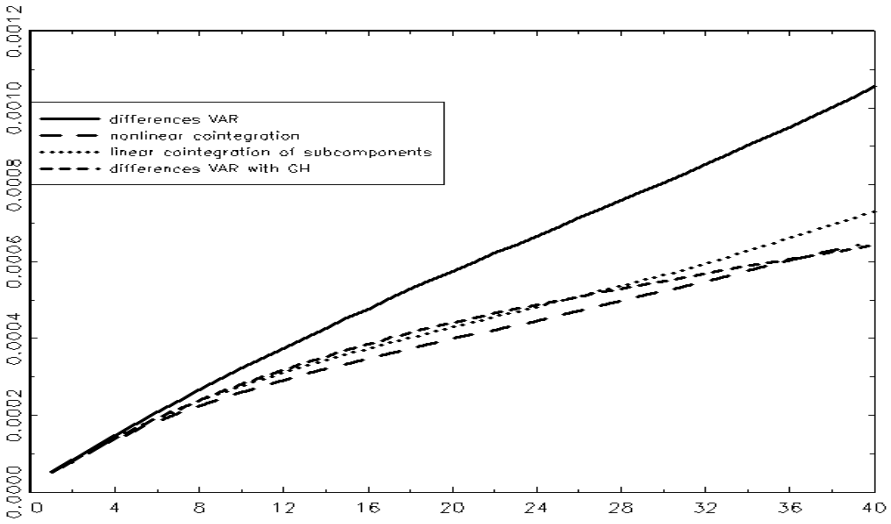

Fig. 5: Trace of the MSE matrix depending on prediction horizon. Generating model is linear cointegration model.

5 Summary and conclusion

The double-stochastic bootstrap evaluation is a logical extension of traditional forecast comparisons that either rely on viewing the observed data as non-stochastic or on non-stochastic prediction. The technique has merits beyond traditional comparisons, as it allows to determine the most 'robust' method among various competing model-based forecasts.

In our application example, such robustness in the sense of the best average performance, no matter which of the considered model actually generated the data, can be attributed to a non-linear co-integration model for the national accounts aggregates.

Another interesting field of application, which we could not study in detail within this paper's limits, was suggested by JUMAH AND KUNST (forthcoming). The outcomes of the bootstrap and the original SN evaluations can be compared. Differences in behavior point to features in the data that cannot be captured by the entertained models. Sometimes, it may make sense to re-consider the original set of prediction models and to attempt to cover these features by creating new prediction structures. In this paper's example, the match of SN and SS evaluations was acceptable but SN graphs tend to be far less regular, with many changes of ranking as the horizon grows.

An additional informal test, which emanates from such a comparison between SN and SS evaluations, concerns the ranking of the various prediction models. Comparable ranking may tend to confirm the generating model class as a good approximation to the DGP. Thus, the experiments of section 4 point to the nonlinear cointegration model as an appropriate choice.

References

[1] A. Escribano and S. Mira, Nonlinear error correction models. *Journal of Time Series Analysis* 23:509–522, 2002.

[2] A. Escribano, Nonlinear Error Correction: The Case of Money Demand in the UK (1878–2000). *Macroeconomic Dynamics* 8:76–116, 2004.

[3] A. Jumah and R.M. Kunst, Modelling Macroeconomic Sub-Aggregates: An Application of Non-Linear Cointegration, *Macroeconomic Dynamics*, forthcoming. Draft version available under http://homepage.univie.ac.at/robert.kunst/macroecdyn1.pdf

[4] H. White, *Estimation, Inference and Specification Analysis,* Cambridge University Press, 1994.

Can a Million Experts improve your Sales' Forecasts ?

Bernard Menezes, Abhishek Seth and Rajveer Singh

Kanwal Rekhi School of Information Technology
IIT Bombay, Powai,
Mumbai-76, INDIA.
Telephone no.:+91-22-25767906
{bernard, abhiseth, rajveer}@it.iitb.ac.in

Abstract: We decompose a time series into three components – trend, seasonality and an irregular component. We apply multiple forecasting methods on each component separately and recombine the forecasts to obtain one for the original series. This results in a very large number of forecasters – its cardinality is multiplicative in the number of forecasters for each component. We study the effect of combining forecasts and compare this approach with model selection. We also experiment with different decomposition methods. We compare our strategy of using decomposition + combining with monthly adaptive Holt-Winter forecasting.

Index Terms: Time series forecasting, Decomposition, Combining Techniques, ARIMA.

1 Introduction

Sales forecasting plays a crucial role in proactive supply chain management – both at the retailer end and further upstream at the distributors, manufacturers and suppliers. Timely and accurate sales forecasts are the key to bridging the gap between supply and demand, thereby decreasing inventory holding costs while maintaining a negligible probability of stock-out. Much work in sales forecasting has centered around the comparison of linear statistical models such as ARIMA [4][5], the Holt-Winter approach (exponential smoothing of level, trend and seasonality) [12] and the use of artificial neural networks [1] [13].

One of the goals of this study is to investigate the effect of a specific form of pre-processing - series decomposition. Recent work in this area [11] confirms our findings [11] that de-trending and de-seasonalization of the data greatly help in improving forecasts. Our work differs from theirs in several ways. In particular, we choose and combine [2][7] a multitude of statistical models to forecast each component series obtained after decomposition.

We perform forecasts for each component series separately. These are then combined to derive the forecast for the original series. The forecasters used here are all statistical - principally ARIMA models of assorted orders. The number of forecasts

for each point is multiplicative in the number of forecasters used for each component. We employ tens of forecasters for each component – this results in several hundred thousand forecasts of each point. We study two approaches to forecasting each point – use the forecaster with the best record from the past and mean/median combining.

The paper is organized as follows. Section 2 deals with forecasting preliminaries. In Section 3, we present and compare multiple decomposition methods used. In Section 4, we report the results of combining forecasts. Section 5 contains our conclusions after a brief comparison with the monthly adaptive Holt-Winter model.

2 Preliminaries

Two of the best known methods of forecasting seasonal data (such as retail sales) are the Holt-Winter method [9] and seasonal ARIMA.

The *Holt-Winter* method uses exponential smoothing of level (S_t), trend (T_t) and seasonal index (I_t) for forecasting the given series X_t. For multiplicative seasonality, the model assumes the form

$$S_t = \alpha(X_t / I_{t-p}) + (1-\alpha)(S_{t-1} + T_{t-1}) \quad (1)$$

$$T_t = \beta(S_t - S_{t-1}) + (1-\beta)T_{t-1} \quad (2)$$

$$I_t = \gamma(X_t / S_t) + (1-\gamma)I_{t-p} \quad (3)$$

Here p is the number of observation points in a cycle (p = 4 for quarterly data). α, β and γ are the smoothing constants. The forecast at time t for time $t+i$ is $(S_t + iT_t)\, I_{t-p+i}$.

The general *multiplicative seasonal ARIMA (p, d, q) X (P,D,Q)* model is expressible as

$$\Phi(B)\, Ø(B^s)(1-B^d)\,(1-B^s)^D X(t) = C_0 + \theta(B)\Theta(B^s)\varepsilon(t), \quad t=1,2,3,....$$

Here C_0 is a constant, ε(t) is a sequence of independent, zero mean and normally distributed errors, d and D are the orders of non-seasonal and seasonal differencing for the time series and $\Phi(B)$, $Ø(B^s)$, $\theta(B)$ and $\Theta(B^s)$ operators are polynomials in B (the backward shift operator) with the following general forms

$$\Phi(B) = 1 - \Phi_1(B) - \Phi_2(B^2) - ... - \Phi_p(B^p) \quad (4)$$

$$Ø(B^s) = 1 - Ø_1(B^s) - Ø_2(B^{2s}) - ... - Ø_P(B^{Ps}) \quad (5)$$

$$\theta(B) = 1 - \theta_1(B) - \theta_2(B^2) - ... - \theta_q(B^q) \quad (6)$$

$$\Theta(B^s) = 1 - \Theta_1(B^s) - \Theta_2(B^{2s}) - ... - \Theta_Q(B^{Qs}) \quad (7)$$

The polynomials $\Phi(B)$ and $\theta(B)$ capture the non-seasonal behavior and $\emptyset(B^s)$ and $\Theta(B^s)$ capture the seasonal behavior of the series. The differencing orders d and D typically have a value *0* or *1*. For non-seasonal ARIMA, $P=D=Q=0$.

Our strategy is to decompose a series into three components. The first two components extract the underlying trend and seasonality in the time series. What remains is ideally random white noise (though in practice it may not be so.) We forecast each component series separately using multiple forecasting models similar to those described above.

A method used in forecasting a component is referred to as an atomic forecaster. Since we decompose the original series into three components, a forecaster for the original series is a triplet made up of the atomic forecasters for each component. The set of such triplets is the Cartesian product of the sets of forecasters for trend, seasonality and the irregular component. We refer to each such triplet of atomic forecasters as an *expert*. Appendix B includes a list of atomic forecasters used in our work. While using an atomic forecaster to forecast a component at point t, we determine the structural parameters of its model by training on the component series up to time t-1.

To forecast a point, we may select an expert with the best track record to date. Specifically, at each point in time, t, we identify the model that yields the best average performance up to time = t-1. We refer to this method as *Best*.

3 Decomposition Strategies

In this section we present various decomposition methods and study their effect on forecasting performance using *Best*.

3.1 Decomposition Methods

There are a number of ways in which the three component series - Trend (*T*), Seasonality (*S*) and the Irregular component (*IC)* - may be defined and combined. We assume that the original series, X, can be obtained from its components using the operators, + and/or ×. We thus have eight ways of combining these series (i.e. $S+T\times I$, $S+T+I$, etc). The pure multiplicative model ($D = T\times S\times I$) seemed to be superior in modeling sales data and so we use it in the experiments reported here.

Let X_t, T_t, S_t and I_t denote the t^{th} point of the respective series. We define the component series below in a manner similar to [10].

$$T_t = \frac{\sum_{i=0}^{11} X_{(t-i)}}{12} \qquad (8)$$

$$S_t = avg(\frac{X_t}{T_t}, \frac{X_{t-p}}{T_{t-p}}, \frac{X_{t-2p}}{T_{t-2p}}, \frac{X_{t-3p}}{T_{t-3p}}, ...) \quad (9)$$

Where p is the seasonality period

$$I_t = \frac{X_t}{T_t \times S_t} \quad (10)$$

We refer to this method of decomposition as D1.

In the above definition of S (Equation 9), all previous years are equally weighted. In many cases, the seasonal component of a series changes with time. It is necessary then to give more importance to the recent past using exponential weights. The seasonal component of decomposition methods D2 – D5 is as below.

$$S_t = \frac{\left(\frac{X_t}{T_t} + \alpha \times \frac{X_{t-p}}{T_{t-p}} + \alpha^2 \times \frac{X_{t-2p}}{T_{t-2p}} + \alpha^3 \times \frac{X_{t-3p}}{T_{t-3p}} + \alpha^4 \times \frac{X_{t-4p}}{T_{t-4p}} \right)}{(1 + \alpha + \alpha^2 + \alpha^3 + \alpha^4)} \quad (11)$$

S_t and T_t are as defined in Equations 8 and 10. The weights $\alpha, \alpha^2, \ldots \alpha^5$, $0 < \alpha <= 1$, are assigned to the de-trended value of X from the past five years (and not to the entire past as in D1) . D2, D3, D4 and D5 differ only in values assigned to α - these are respectively 0.3, 0.6, 0.8 and 1.

3.2 Experiments and Results

For all our experiments, we used a total of 31 series (25 real + 6 synthetic) representing monthly sales. The series are listed in Appendix A. In all cases, the initial 30 points are used exclusively for training. Forecasting performance is measured using MAPE (Mean Absolute Percentage Error) starting with the 31st point of the series. Also, in all cases, models are re-trained (i.e., their parameters are re-computed) after performing a forecast.

The goal of our first experiment was to compare the forecasting performance with and without decomposition. For the latter, we selected 26 forecasting models. These are mostly seasonal ARIMA or variations of the Holt-Winter model (Appendix B). In the case of decomposition, we employed the services of 86, 34 and 33 atomic forecasters for *T*, *IC* and *S* respectively thus yielding a total of 96492 experts. The names and notations for the atomic forecasters are listed in Appendix B.

Table 1 shows the MAPEs using *Best*. The WoD column shows the MAPEs without performing decomposition. Columns D1-D5 show the MAPEs using the different decomposition methods. The MAPEs using decomposition are better than the MAPEs

without decomposition in a majority of the series. MAPEs that are within 1% of the best MAPE in a given row are shown in bold. It is clear that some decompositions are better suited to some series than others. Every decomposition method performs best for at least one series while no single method emerges a clear winner across all series.

The last row shows the improvement using decomposition over the case without decomposition averaged over all series – the greatest improvement is in the case of D1 which is 3.65% better than the MAPE without decomposition. D1 also performs best in about 50% of the series but it is also the worst performer in series such as Spk and Sto.

Series	WoD	D1	D2	D3	D4	D5
Abr	2.94	**2.76**	2.84	2.8	2.83	**2.75**
Beer	2.47	**2.23**	2.44	2.39	2.41	2.33
Clo	2.5	2.47	**2.32**	2.39	2.37	2.44
Dry	**8.44**	8.6	8.58	**8.47**	**8.43**	8.61
Eqp	0.75	0.8	0.77	0.77	**0.74**	0.81
For	8.17	7.98	8.14	7.78	**7.65**	7.89
Fur	2.3	2.43	**2.25**	2.29	**2.26**	2.29
Gas	2.6	**2.47**	2.62	2.67	2.52	**2.47**
Gro	1.39	**1.18**	1.37	1.27	1.25	1.26
Hsa	8.33	**7.95**	8.37	8.15	8.29	8.35
Jew	3.78	3.96	3.76	3.75	**3.65**	3.77
Mer	0.83	**0.8**	**0.8**	**0.81**	0.82	0.83
Mot	**1.47**	1.52	1.63	1.62	1.57	1.58
New	4.35	**3.99**	4.14	**4.01**	4.04	4.16
Pap	7.33	**5.16**	5.38	**5.18**	5.25	5.24
Red	9.83	9.7	**9.05**	9.26	9.38	9.35
Ros	14.04	**13.67**	13.94	**13.55**	14.1	14.25
Sho	3.04	3.07	**2.97**	**2.99**	3.19	3.06
Sof	3.97	3.57	3.71	**3.53**	3.58	3.6
Spa	8.25	**7.94**	8.38	8.24	8.23	8.07
Spk	**12.71**	13.46	**12.6**	12.78	12.97	13.22
Sto	0.9	0.97	**0.85**	0.89	0.91	0.93
Swe	16.85	16.05	**15.67**	**15.74**	15.93	16.37
Tot	0.66	0.66	0.69	0.66	0.65	**0.64**
Win	8.13	**7.41**	7.77	7.73	7.72	7.57
S1	1.66	**1.62**	**1.62**	1.65	1.67	1.68
S2	3.42	3.11	3.28	3.62	3.11	**3.07**
S3	3.06	**2.33**	2.49	2.53	2.48	2.36
S4	**0.61**	0.65	0.65	0.63	0.66	0.67
S5	**2.90**	**2.89**	2.96	2.94	2.94	2.96
S6	5.98	**5.68**	6.03	6.09	5.94	5.87
%Impr	0	3.65	2.59	3.03	3.34	2.92

Table 1 MAPES using various decomposition methods in conjunction with *Best*

4 Combining

The *Best* method outlined in Section 2 suffers from the problem of model selection instability [2] – a model's superior overall performance in the past is not necessarily

indicative of excellent performance in the future. To reduce the risk of making a poor choice, we resort to combining the forecasts of multiple experts. The most obvious option is to compute the mean or median of all the experts' forecasts. We experimented with combining in both scenarios – without decomposition (using 26 experts) and with decomposition (five decomposition methods, each using 96,492 experts.) By way of clarification, we now have six possibilities:

1. No Decomposition
   a) Using *Best* (26 forecasters)
   b) Mean of 26 forecasters
   c) Median of 26 forecasters
2. Decomposition (5 methods)
   a) Using *Best* (96,492 * 5 experts)
   b) Mean of 96,492 * 5 experts
   c) Median of 96,492 * 5 experts

The MAPES for each of the above cases is shown in Table 2. The last row shows the average percentage improvement of the Mean/Median approach over *Best*. It is clear that, overall, there is no advantage of combining over *Best* when no prior decomposition is employed. However, with decomposition, the median of all experts' forecasts exhibits better performance over *Best* on the average. Using the median rather than the mean yields an improvement in the vast majority of cases especially when decomposition is not employed.

	WOD			WD		
Series	*Best*	MN	MED	*Best*	MEAN	MED
Abr	**2.94**	3.44	3.4	**2.8**	2.95	2.96
Beer	**2.47**	2.66	2.63	**2.22**	2.5	2.48
Clo	2.5	2.66	**2.44**	2.39	2.48	**2.29**
Dry	8.44	8.61	**8.03**	8.69	8.4	**8.27**
Eqp	**0.75**	1.15	**0.75**	0.82	0.79	**0.75**
For	**8.17**	8.29	8.29	8.09	**7.74**	**7.74**
Fur	2.3	2.25	**2.16**	2.28	2.35	**2.13**
Gas	**2.6**	3.31	2.65	2.62	2.62	**2.56**
Gro	**1.39**	1.54	1.62	**1.18**	1.71	1.29
Hsa	**8.33**	9.08	**8.31**	**7.75**	8.25	8.14
Jwl	**3.78**	4.56	**3.79**	3.63	3.67	**3.52**
Mer	0.83	0.99	**0.78**	0.83	0.78	**0.75**
Mot	**1.47**	2.03	1.51	**1.59**	1.75	1.69
New	4.35	**3.93**	**3.97**	4.14	**3.93**	**3.94**
Pap	7.33	6.25	**5.91**	5.17	5.12	**5.02**
Red	**9.83**	10.22	**9.77**	**9.33**	9.44	**9.33**
Ros	**14.04**	14.71	14.69	13.88	**12.86**	**12.86**
Sho	**3.04**	3.18	3.1	3	**2.93**	**2.91**
Sof	3.97	3.94	**3.57**	3.57	**3.39**	**3.35**
Spa	**8.25**	9.22	8.68	**8.17**	8.31	8.43
Spk	12.71	12.67	**11.95**	13.24	**12.45**	**12.38**
Sto	0.9	1.06	**0.84**	0.9	0.85	**0.83**
Swe	16.85	16.23	**15.59**	15.98	**15.33**	15.49
Tot	0.66	0.94	**0.64**	**0.67**	0.73	0.69
Win	8.13	8.21	**7.89**	**7.41**	7.6	7.52
S1	**1.66**	1.91	1.83	**1.65**	1.77	1.68
S2	3.42	3.38	**3.14**	**3.24**	3.37	**3.27**
S3	3.06	3.20	**3.01**	**2.41**	2.90	2.79
S4	**0.61**	1.47	0.85	0.66	**0.62**	0.63
S5	**2.90**	3.44	3.22	2.95	**2.89**	**2.88**
S6	**5.98**	6.39	6.41	**5.84**	**5.79**	5.86
%Impr	0	-14.58	-0.97	0	-2.32	0.87

Table 2 MAPES using *Best*, Mean/Median with and without decomposition

5 Discussion and Conclusions

In addition to comparing the schemes presented here with one another, we also compared them to results using the Holt-Winter model. For the latter, we used a monthly adaptive approach where the smoothing constants are updated each month. To forecast the value of X at point t, a global search is conducted to determine the values of α, β and γ (Equations 1-3) that minimize the MAPE up to and including time = t-1. The Holt-Winter set of equations with the updated smoothing constants are then used to forecast X at time = t.

Table 3 shows that *Best* without decomposition performs better than HW in 20 of the 31 series while *Best* after decomposition performs better in 26 series. Decomposition followed by combining using the median performs better than HW in 29 series, its MAPE being only marginally higher in the other two series. The last row of Table 3 shows the average percentage improvement over HW across all series considered here. As can be seen, decomposition greatly helps - the improvement using the naïve median approach (9.08% improvement) is greater than that using *Best* (7.88% improvement). *Best* without decomposition is also better than HW but the improvement in this case is only about 4%.

		Best		MED
Series	HW	WoD	D	D
Abr	3.26	2.94	**2.8**	2.96
Beer	2.52	2.47	**2.22**	2.48
Clo	2.46	2.5	2.39	**2.29**
Dry	**8.26**	8.44	8.69	**8.27**
Eqp	0.97	**0.75**	0.82	**0.75**
For	7.9	8.17	8.09	**7.74**
Fur	2.37	2.3	2.28	**2.13**
Gas	2.7	2.6	2.62	**2.56**
Gro	1.4	1.39	**1.18**	1.29
Hsa	10.17	8.33	**7.75**	8.14
Jwl	3.76	3.78	3.63	**3.52**
Mer	0.85	0.83	0.83	**0.75**
Mot	2.21	**1.47**	1.59	1.69
New	4.13	4.35	4.14	**3.94**
Pap	5.19	7.33	5.17	**5.02**
Red	9.57	9.83	**9.33**	**9.33**
Ros	15.12	14.04	13.88	**12.86**
Sho	3.02	3.04	3	**2.91**
Sof	3.86	3.97	3.57	**3.35**
Spa	8.39	**8.25**	**8.17**	8.43
Spk	12.79	12.71	13.24	**12.38**
Sto	0.92	0.9	0.9	**0.83**
Swe	15.8	16.85	15.98	**15.49**
Tot	0.92	**0.66**	0.67	0.69
Win	7.55	8.13	**7.41**	7.52
S1	1.73	**1.66**	**1.65**	1.68
S2	4.00	3.42	**3.24**	**3.27**
S3	3.79	3.06	**2.41**	2.79
S4	0.77	**0.61**	0.66	0.63
S5	2.95	**2.90**	2.95	**2.88**
S6	6.15	5.98	**5.84**	**5.86**
%Impr	0	3.98	7.88	9.08

Table 3 Improvements of various methods over HW

Decomposing a series into its trend, seasonality and other components seems like a "natural" top-down approach to forecasting. *Best* and combining, both, reap considerable benefit if the time series is first decomposed. Thanks to decomposition and the use of multiple forecasters for each component, we have effectively increased the cardinality of the pool of experts (26 forecasters without decomposition and 96492 using decomposition). Using 5 decomposition schemes further magnifies the pool of experts.

We report results using combining obtained from the median of all experts' forecasts. Earlier work in the area of combining indicates that the mean or median performs as well as more sophisticated approaches using weighted experts, etc. Most earlier studies, however, have far fewer forecasters, usually less than hundred. It is yet to be established whether the mean/median approach still wears the crown when the number of experts touches a million. Our current work in this area seems to indicate otherwise.

References

[1] I.Alon, M. Qi and R.J.Sadowski: "Forecasting aggregate retail sales: a comparison of artificial neural networks and traditional methods". *Journal of Retailing and Consumer Services* 8, pp. 147-156, 2001.

[2] J. S. Armstrong. "Combining forecasts: The end of the beginning or the beginning of the end?" *International Journal of Forecasting*, 1990.

[3] J. Scott Armstrong, Fred Collopy. "Error Measures for Generalizing about forecasting methods: Empirical comparisons" *International Journal of Forecasting*, 8, pp. 69-80, 1992.

[4] G.Box, G.Jenkins and G.Reinsel "Time Series Analysis – Forecasting and Control" Third Edition, Prentice Hall, 1994, Englewood Cliffs, NJ.

[5] P. J. Brockwell and R. A. Davis. "Introduction to Time Series and Forecasting". Springer, 2001.

[6] C-W. Chu and G.P.Zhang "A comparative study of linear and non-linear models for aggregate retail sales forecasting". *International Journal of Production Economics,* 2003.

[7] R. T. Clemen. "Combining forecasts: A review and annotated bibliography". *International Journal of Forecasting*, 1989.

[8] G.Gangadharan, L.Khandelwal, and B.L.Menezes, "Forecasting in Centralized Supply Chain Management", *Smart E-Business International Workshop* (part of First Indian International Conference on Artificial Intelligence), Hyderabad, pp. 95-107, Dec. 20, 2003.

[9] L.Krajewski and L.Ritzman, "Operations Management – Strategy and Analysis". 5th Edn, Addison Wesley Publishing Company, 1999.

[10] R.Pindyck and D.L.Rubinfeld, "Econometric Models and Economic Forecasts", 4th Edition McGraw Hill International, 1998.

[11] Min Qi G. Peter Zhang. "Neural network forecasting for seasonal and trend time series". *European Journal of Operational Research*, pp. 501–514, 2005.

[12] R.Snyder, A.Koehler, J.K.Ord, "Forecasting for inventory control with exponential smoothing". *International Journal of Forecasting*, 18, pp. 5-18, 2002.

[13] Frank M. Thiesing and Oliver Vornberger. "Forecasting sales using neural networks". *Proceedings of the International Conference on Computational Intelligence, Theory and Applications*, pp. 321–328, April 1997.

[14] Time Series Data Library. Website: http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/

[15] Economic Time Series. Website: http://www.economagic.com/cenret.htm

[16] P. Gulhane, B. Menezes, T. Reddy, K. Shah, S.A. Soman (2005). "Forecasting using decomposition and combining of experts". *International Conference on Artificial Intelligence*, Las Vegas, Nevada.

Appendix

A. Time Series Used

For our experiments, we have used real world sales series from [14] and [15]. These series are described in Table 4 and 5. We also considered a few synthetic series with the following component definitions. Series S4, S5 and S6 are borrowed from [11].

$$A(t) = 500 + 4t; \quad 1 \le t \le 200 \tag{12}$$

$$B1(t) = 1 + 0.2\sin(2.\pi.t/12) \tag{13}$$

$$B2(t) = 1 + 0.3\sin(2.\pi.t/12) \tag{14}$$

$$C(t,x) = Normal\ (0,x) \tag{15}$$

The series that we have synthesized are:

- S1: Here D(t) = A(t) * B1(t) * C(t,0.02)
- S2: Here D(t) = A(t) * B3(t) * C(t,0.02), where B3(t) is defined as

$$B3(t) = B1(t)\ \ if\ \ \mathrm{mod}(\lfloor t/24 \rfloor) = 0;\ B2(t)\ \ otherwise; \tag{16}$$

- S3: It is created as D(t) = A(t) * B4(t) * C(t,0.02), where B3(t) is defined as

$$B4(t) = B1(t)\ \ if\ \ \mathrm{mod}(\lfloor t/12 \rfloor) = 0;\ B2(t)\ \ otherwise; \tag{17}$$

Series	Description	Series	Description
Abr	Gasoline demand Ontario	Dry	Australia sales of dry white wine
For	Australian sales of fortified wine	Hsa	Sales of new houses sold in the USA
Pap	Sales of printing and writing paper	Red	Australian sales of red wine
Ros	Australian sales of rose wine	Spa	CFE specialty writing papers
Spk	Australian sales of sparkling wine	Swe	Australian sales of sweet white wine
Win	Australian wine sales		

Table 4 Series from Time Series Library

Beer	US Retail Sales: beer, wine, liquor stores	Clo	US Retail Sales: Clothing stores
Eqp	US Retail Inventories: Building materials	Fur	US Retail Sales: Furniture stores
Gas	US Retail Sales: Gasoline stations	Gro	US Retail Sales: Grocery stores
Jwl	US Retail Sales: Jewelry stores	Mer	US Retail Inventories: General merchandise
Mot	US Retail Inventories: motor vehicle dealers	New	US Retail Sales: New car dealers
Sho	US Retail Sales: Shoe stores	Sof	US Retail Sales: Computer and software
Sto	US Retail Inventories: Department stores	Tot	US Retail Inventories: totals

Table 5 Series from Economic Time Series

B. Forecasting Models Used

Table 6 shows the forecasters used for the original series and for T, S and I. In the *Type* column, the notation (p,d,q) and (p,d,q)(P,D,Q) are standard notation for ARIMA and Seasonal ARIMA models respectively. Most of the atomic forecasters are parsimonious ARIMA or seasonal ARIMA models (p <= 3 and q <= 2). Options such as differencing (d = 1) and log transformations are also used.
The numbers in the *Var* column have the following meaning:

- 1 No log transformation.
- 2 Log transformations.
- 3 No log transformation and $\mu = 0$.
- 4 Log transformations and $\mu = 0$.

Here μ is the parameter of the Seasonal ARIMA in:

$$(1-B)^d(1-B^s)^D Y_t = \mu + \frac{\theta(B)\theta_s(B^s)}{\phi(B)\phi_s(B^s)}a_t \quad (18)$$

Table 6 Forecasting Models selected for series and various components

Type	Var	Type	Var	Type	Var
Trend Experts		**Irregular Experts**		**WoD Experts**	
([0-2],1,1)	1,2,3,4	Mean		(0,0,1)(0,1,1)s	4
([0-2],1,2)	1,2,3,4	([0-1],1,0)	1,2	(0,1,0)(0,1,1)s	4
([0-2],2,1)	1,2,3,4	([1-2],0,0)	1,2	(0,[1-2],2)(0,1,1)s	4
([1-2],1,0)	1,2,3,4	(1,1,2)	1,2	(3,0,0)(1,1,0)s	4
(1,2,0)	1,2,3,4	(0,1,1)	1	([1-2],0,0)(0,1,1)s	3
([0-1],1,0)(1,0,0)s	1,2,3,4	(0,0,0)([0-1],0,1)s	1,2	(2,1,0)(0,1,0)s	3
([0-1],1,0)([0-1],0,1)s	1,2	([0,2],0,0)(1,0,0)s	1,2	(2,1,2)(0,1,1)s	3
(1,1,[1-2])(0,0,1)s	1,2	(0,1,1)(1,0,0)s	3,4	(0,1,1)(0,1,1)s	2,3,4
(1,1,2)(1,0,0)s	1,2	(0,0,0)(0,1,1)s	3,4	(0,0,0)(0,1,1)s	2,3,4
(3,1,0)(0,0,1)s	1,2	Linear Exp with		(0,0,0)(0,1,2)s	2
([1-2],0,1)	1	AR=1,2,3	1,2	(0,1,0)(1,0,0)s	1
(3,1,0)	1,2	Linear Exp	1,2	(0,0,0)(1,1,0)s	3,4
(2,1,0)(1,0,0)s	1,3	(3,0,0)(1,0,0)s	1	(2,1,0)(0,1,1)s	3,4
(3,1,0)(1,0,0)s	1,2,3	(0,0,0)(3,1,1)s	4	(3,0,0)(0,1,1)s	1,3
(2,0,0)(1,0,0)s	2,4	**Season Experts**		Seasonal Exp Smoot	1,2
(0,1,1)(1,0,0)s	3,4	([0-3],0,0)(0,1,1)s	2,4	Add Winters	
Holt Method		([0-1],0,1)(0,1,1)s	2,4		
		([1-3],1,0)(0,1,1)s	2,4		
		([0-2],1,1)(0,1,1)s	2,4		
		([0-2],1,2)(0,1,1)s	2,4		
		(0,0,2)(0,1,1)s	2,4		
		Holt Winter			

Using autoregressive estimators for data with missing samples.

Alois Schlögl[1,2]

[1] Fraunhofer, FIRST-IDA,
Kekulestraße 7, 12489, Berlin, Germany
[2] University of Technology Graz Institute for Human-Computer Interfaces
Krenngasse 37, A-8010 Graz, Austria

Abstract. Autoregressive models have been widely used for time series prediction. Several univariate and multivariate estimators have been implemented for the use with missing data. These algorithms were used to perform the prediction of the ESTSP2007 competition. In order to determine the free parameters (model order and estimation algorithm), segments of the competition data were encoded as missing samples; then the prediction error on these samples was used for comparison. The predication using different models are shown. The prediction result of an AR(p=300) process with superimposed sinusoid and a constant mean term was submitted to the ESTSP2007 prediction competition.

1 Introduction

Time series analysis is an important approach for analyzing biomedical signals like the electroencephalogram (EEG). The first autoregressive estimation of EEG can be traced back to Lustick et al. Fenwick et al. Zetterberg and Gersch in the late 1960s. Nowadays, multivariate autoregressive (MVAR) models are used to estimate various coupling measures, e.g. Directed transfer function, Coherence, Partial Directed Coherence and several other coupling parameters (Schlögl and Supp, 2006).

All these methods relay on algorithms for estimating autoregressive model parameter. Several estimators for univariate (Levinson-Durbin algorithm, the Burg algorithm and the geometric lattice algorithm) and multivariate (Nutall-Strand, Vieira-Morf and Multichannel Yule-Walker) autoregressive models have been implemented for Octave and Matlab, and were made available under the GNU General Public License through the TSA-toolbox (Schlögl 1996-2006).

Recently, a comparison of these MVAR estimators and ARFIT (Schneider and Neumeier, 2001) was performed (Schlögl 2006). Because, biomedical signals are often contaminated by artifacts (causing missing samples), the estimators in the "TSA-toolbox for Octave and Matlab" were modified to enable the handling of data

with missing samples, the missing samples have to be encoded as not-a-number (NaN). In this work, the cross-validation scheme for determining the free parameters (model order, choice of estimation algorithm) will utilize this property of being able to handle data with missing samples.

This work describes a contribution to the Time Series prediction competition 2007, using a simple autoregressive model. Several different AR estimators will be used, the degrees of freedom (model order and choice of estimation algorithm) will be determined by a cross-validation procedure on the training data. It is not expected that a pure autoregressive method will provide the optimal performance because detailed data inspection suggests that the data is non-Gaussian and contains at least one sinusoidal component. Therefore, the most prominent components are identified, and only the residual process is modeled by an autoregressive model.

2 Method

2.1 Initial analysis

Visual inspection of data y

investigating the data, as well a the limited number of data samples, only the second order statistical properties are investigated. For this purpose, an autoregressive model can be used. An AR process is defined as x

by Wiggens and Robbinson, Nutall, Strand, Vieira and Morf (Marple, 1987, Schloegl, 2006). In this work, two univariate AR estimators (Levinson-Durbin recursion and Burg) and six MVAR estimators (Nutall-Strand, Vieira-Morf and Levinson-Wiggens-Robbinson recursion, each using "biased" and "unbiased" covariance estimation) were used. The algorithmic implementation was provided by DURLEV.M, LATTICE.M and MVAR.M from the TSA toolbox for Octave and Matlab (Schlögl 1996-2006).

These algorithms were modified in such a way that data with missing samples (encoded as NaN) can be handled. The modification for handling missing samples is based on the idea that missing samples are ignored (skipped) when estimating some expectation values, a counter counts the number of valid terms.

Specifically, the estimation of the expected autocorrelation r

3 Results

3.1 Decomposition of the data

Based on the initial analysis using visual inspection and histogram analysis, a constant mean and a sinusoidal component have been identified. After subtraction of these components, a residual process is obtained. The various components are shown in Figure 1.

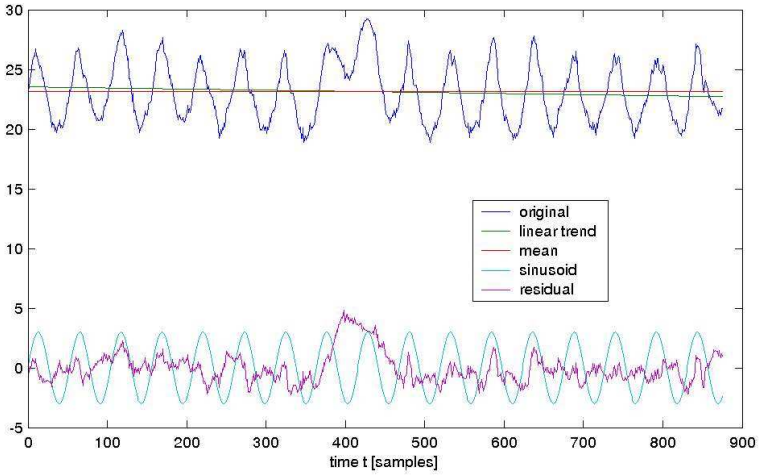

Figure 1: Decomposition of the data. The time series is decomposed into a (constant) mean, a sinusoid with amplitude A=3 and a cycle duration of 51.9 samples, and the residual activity.

3.2 Selecting the estimator and the order of the AR model

Figure 2 shows the mean square of the predication error for different estimators and model orders. The prediction error of all estimators were computed up to moder p=100. Then, it became apparent that several estimators perform significantly worse. Therefore, only the univariate Levinson-Durbin and the Burg methods as well as the MVAR1 method (Levinson-Wiggins-Robinson recursion) were used for higher model orders.

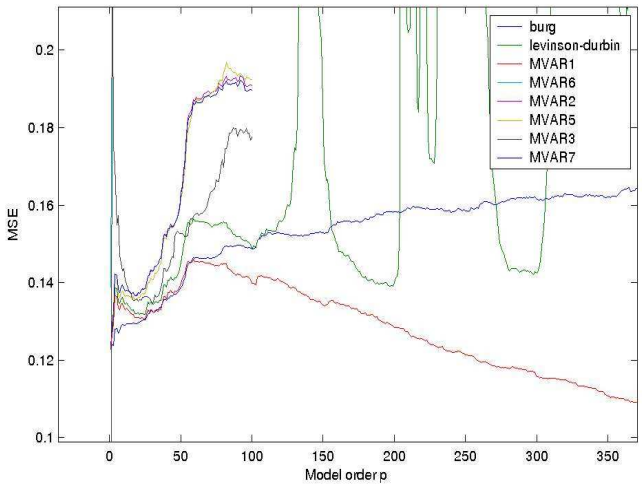

Figure 2: Comparison of different model estimates. Most algorithms were computed only to model order p=100, because this seem to perform worse. Only MVAR1, Levinson-Durbin and Burg were computed for larger model orders, too.

3.3 **Prediction results**

The prediction of 4 different models is shown in Figure 3. Three models are based on sole AR models, using different estimators and different model orders; two models use the decomposition model and the residual is modeled by AR processes. The submitted prediction for the ESTSP2007 competition was based on the decomposition model combined with an residual AR process estimated with an MVAR1(p=300) model.

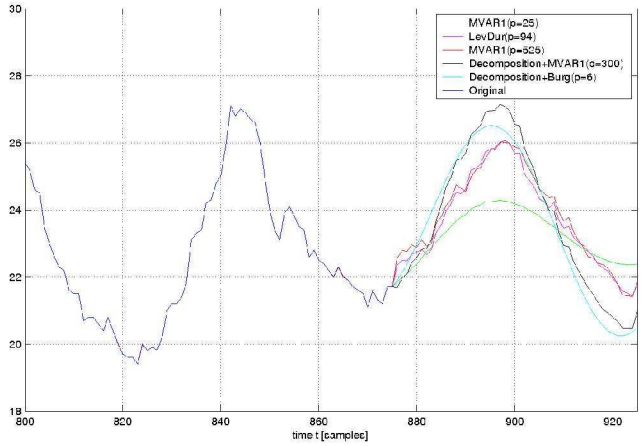

Figure 3: Prediction of different models.

4 Conclusion

Several standard AR estimators (including Levinson-Durbin, Burg, LWR, Nutall-Strand, Vieira-Morf) were modified for its use on data with missing samples. The solution follows the general idea of estimating the expectation values from the available samples only.

The computational effort of the AR estimators increased at most by a factor of 2. This is much less then the computational effort of maximum likelihood estimators or imputation methods. In this work, missing samples were used for the implementation of the cross-validation procedure. Moreover, it is shown that estimators are stable also for large model orders.

The fact that the estimators are able to handle data with missing sample values simplified the implementation of the a cross-validation procedure. The cross-validation was used to exclude bad-performing estimators and demonstrate the fact that large model orders need to be used for this data. However, due to the limited time, submission for the ESTSP 2007 competition has based on subjective judgment using an AR mode of p=300.

References:

[1] A. Schlögl and G. Supp, Analyzing event-related EEG data with multivariate autoregressive parameters. in: Progress in Brain Research: *Event-related Dynamics of Brain Oscillations. Analysis of dynamics of brain oscillations: methodological advances*. 159:135 – 147, Elsevier, 2006.

[2] A. Schlögl (1996-2006) TSA-Toolbox for Octave and Matlab. http://octave.cvs.sourceforge.net/octave/octave-forge/extra/tsa/

[3] S.L. Marple, 1987, *Digital Spectral Analysis with Applications.* Prentice Hall, Englewood Cliffs, NJ.

[4] T. Schneider and A. Neumaier, Algorithm 808: Arfit – a Matlab package for the estimation oparameters and eigenmodes of multivariate autoregressive models. ACM T Math. Softwaer 27:58-65, 2001.

[5] A. Schlögl, Comparison of Multivariate Autoregressive Estimators. *Signal processing, 86(9): 2426-9, Elsevier,* 2006.

Removing seasonality on time series, a practical case

L. J. Herrera, H. Pomares, I. Rojas, G. Rubio, A.Guillén

University of Granada - Dept of Computer Architecture and Computer Technology
Granada- Spain

Abstract. A good analysis of a time series is essential in order to obtain optimal predictions, specially when performing long term forecasting. Patterns in time series are normally described in terms of two basic components: trend and seasonality. There are a number of techniques in order to identify and eliminate those components from time series; however there is no automatic techniques and each specific problems needs a detailed analysis. This paper presents a solution to the benchmark proposed for the ESTSP'2007, obtained by exploiting the seasonality and trend properties of the provided time series.

1 Introduction

Time series forecasting is a challenge in many fields. There exist many techniques that are applied to the problem of predicting new values in univariate time series [1]. Among them we may enumerate linear methods such as ARX, ARMA, etc. and nonlinear ones such as artificial neural networks, fuzzy models, support vector machines, etc. However, it is important to perform an appropriate time series analysis before tackling with the modeling of the time series, in order to discover possible patters to help in the prediction. This becomes even more essential on long term prediction problems, in which the uncertainty increases with the horizon of prediction [4].

Most time series patterns can be described in terms of two basic classes of components: trend and seasonality. Trends represent a general systematic linear or nonlinear component that changes over time and does not repeat or at least does not repeat within the time range captured by the data available. Seasonality represents a certain pattern that repeats itself in systematic intervals over time [5]. Those two general classes of time series components may coexist in real-life data. An example would be the sales of a company that can rapidly grow over years, but that still follow a consistent seasonal pattern.

This work presents a practical study on the analysis and model of a time series: the benchmark proposed in the European Symposium on Time Series Prediction (ESTSP'2007). The series is analyzed, and the trend and seasonality are eliminated. Finally some improvements are proposed by modeling the obtained series using a simple AR model [2, 6].

The rest of the work is structured as follows. Section 2 briefly describes the ESTST'2007 benchmark time series. Section 3 performs an analysis of the series, describing its trend and seasonality components. Section 4 explains how new values of the series are predicted. Finally section 5 concludes the paper.

2 ESTSP'2007 Benchmark

The data set provided is shown in *fig.* 1. The number of samples in the time series is 875. The goal of the competition is the prediction of the 50 next values of the time series. The evaluation of the performance will be done using the mean squared error MSE obtained from the prediction of both the 15 and the 50 next values.

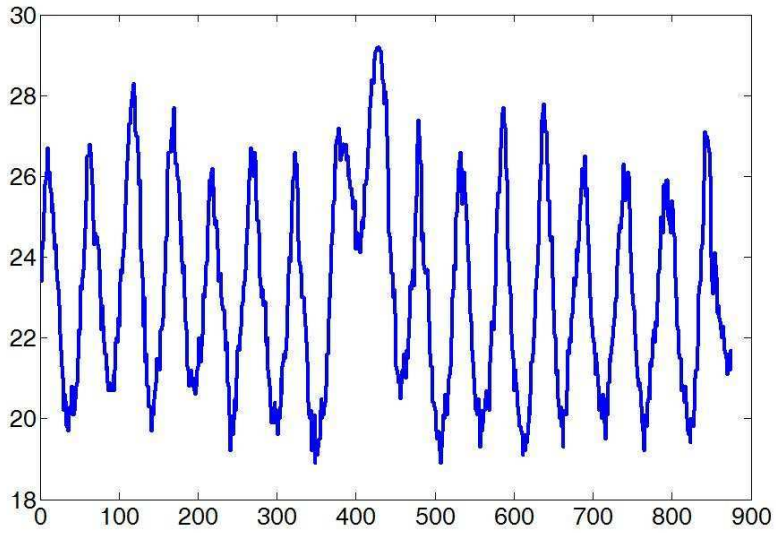

Fig. 1: Original data set

In the tests performed in this work, from the 875 data samples of the series $x(t)$, the first 750 data samples were used as training data, leaving the remaining 125 data samples as test data.

3 Data Analysis

The first characteristic that is noticed in the series is that it shows to have seasonality. However, it is to be noticed too that when performing a input/output (I/O) modeling of the time series in the form

$$\hat{x}(t+h) = F\left(x\left(t-0\right),\ x\left(t-1\right), \ldots\ x\left(t-\tau\right)\right) \tag{1}$$

it might be not convenient to work in the original series domain, since it is possible that some fluctuation of the series (like the peak within the 400-450 samples) are not discovered -there is no I/O data available from which to learn those fluctuations. That is to say, the trend of the time series needs to be taken into account explicitly.

3.1 Differenced Time Series

This will be done, as is proposed in [2] by differencing the time series. The differenced series with lag 1 is calculated as [3]

$$d(t) = x(t+1) - x(t) \tag{2}$$

The differenced time series is shown in *fig.* 2. The number of samples in the series is now is 874. In the differenced series the trend has been eliminated, although the resulting series isn't still stationary. The seasonality will be also eliminated as it is explained in the next subsection.

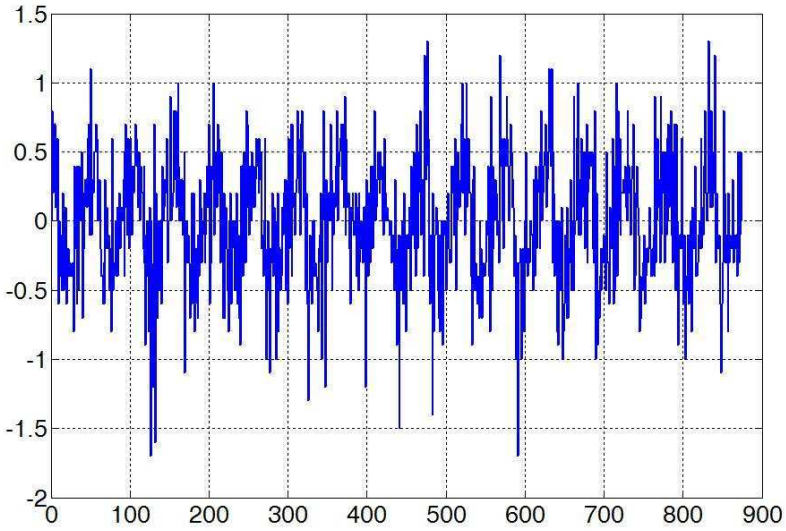

Fig. 2: Differenced time series

3.2 Seasonality in Time Series

Seasonality in time series occurs when there is a certain pattern that is repeated each k elements. The data series in *fig.* 2 shows to have a certain seasonality, as can be seen from the autocorrelation function of the time series in *fig.* 3.

The period of the seasonality can be obtained by a number of techniques [2]. In this work, the period has been obtained by evaluating the distances between the different supposed-seasons of the differenced time series. The specific period for which the distances among the different seasons was lowest, was taken as a solution. The specific period found in this case was 52; *fig.* 4 shows the superposition of the different seasons of 52 samples of the time series. This value reminds a weekly registered phenomena along different years (seasons).

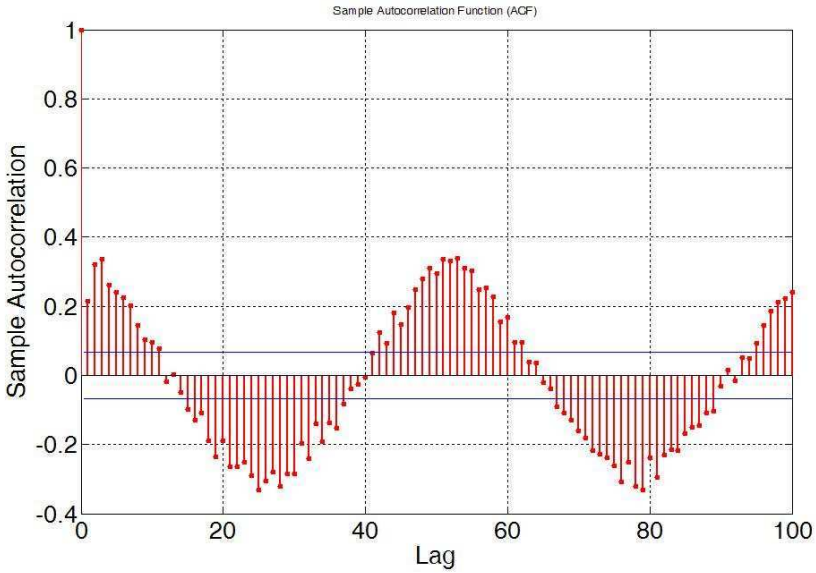

Fig. 3: Sample Autocorrelation Function of the Differenced Series

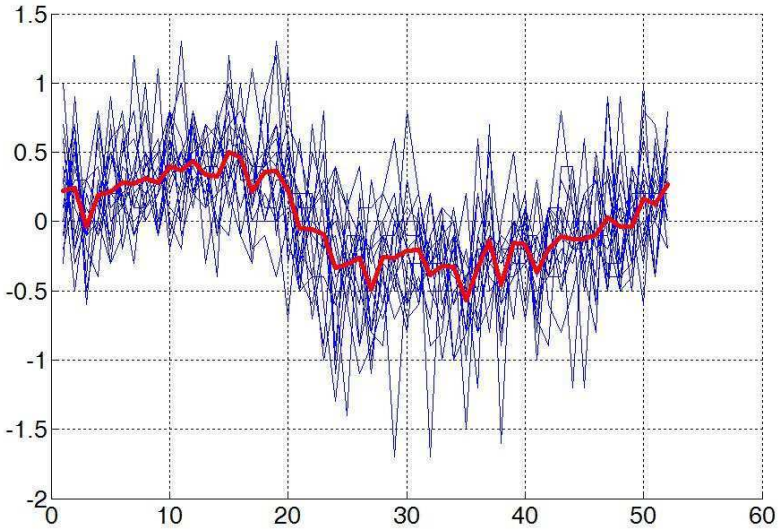

Fig. 4: Joint representation of 16 complete seasons in the differenced time series. The bold line represents the mean value for each time step of a season of 52 values

Removing the seasonality in the series was performed by calculating the average values on a season, and by subtracting these average values to the series. *fig.* 4 shows in bold the mean of the differences series values for a season. Thus the resulting time series will be the difference of each time step of the season with respect to its mean value for the sequence of 16,8 seasons in the 874 differenced series samples. *Fig.* 5 shows the time series after removing the average values within each season.

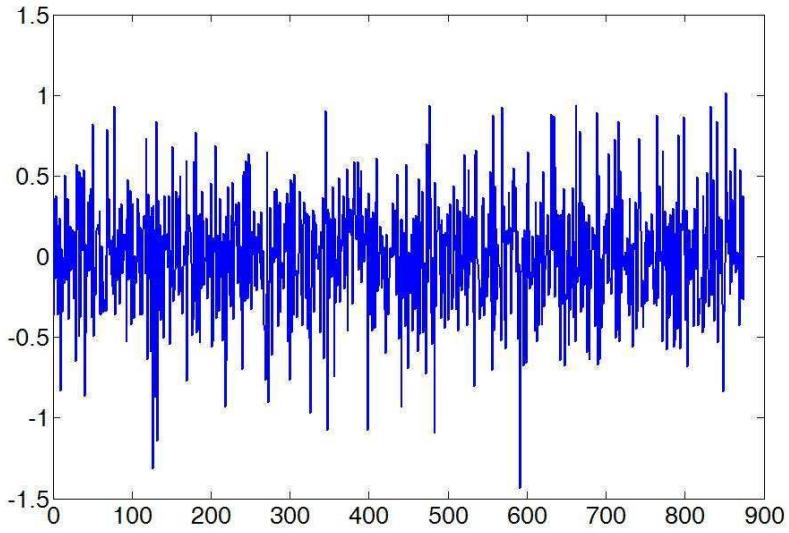

Fig. 5: Differenced series after erasing seasonality, $u(t)$

4 Prediction of Future Values in the Time Series

Once the series has been differenced and the seasonality has been removed from the series, the series can be modeled.

Any paradigm or model could be used to perform the modeling of the series $u(t)$ in *fig.* 5

$$\hat{u}(t+h) = F\left(u\left(t-0\right),\ u\left(t-1\right), \ldots\ u\left(t-\tau\right)\right) \tag{3}$$

where

$$u(t) = d(t) - meanValue(t) \tag{4}$$

However, a first analysis of the series shows that the series is very noisy, and there are few chances to perform any prediction. The autocorrelation function of

the series in *fig.* 6 shows that just the $u(t)$ brings information about an objective variable to predict $u(t+1)$.

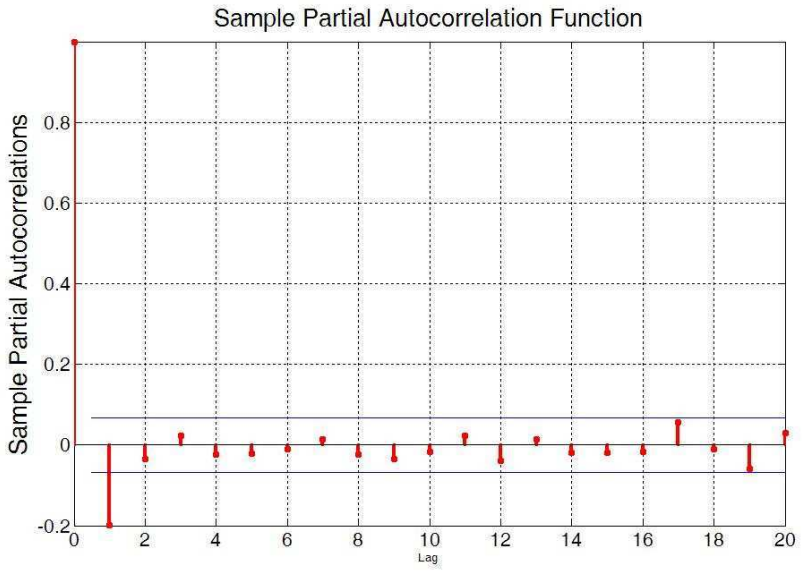

Fig. 6: Sample partial autocorrelation function of the series $u(t)$

Therefore a simple AR model was built to predict the value $u(t+1)$ as a function of $u(t)$; the model found as optimal parameter (being the mean of the series equal to 0)

$$\hat{u}(t) = u(t-1) * (-0.2123) \tag{5}$$

However due to the low performance of the model, the AR system was used to predict just the first 5 values of the series: $\hat{u}(876) : \hat{u}(880)$ (with an estimated NRMSE of almost 1).

The final prediction was obtained by adding the predicted $\hat{u}$ to the expected mean values obtained by un-seasoning the differenced series (see *fig.* 4). After the series was reconstructed by reversing the differencing procedure.

Fig. 7 shows the prediction for the desired data points for the ESTSP'2007 competition. The expected MSE of the prediction procedure described in this paper is 0.81, that has been obtained by using the first 750 data points as training (to obtain the average of the differenced series for each season) and the rest for testing complete predictions of 50 data points. The final prediction was obtained by using the whole 875 data samples of the series as training.

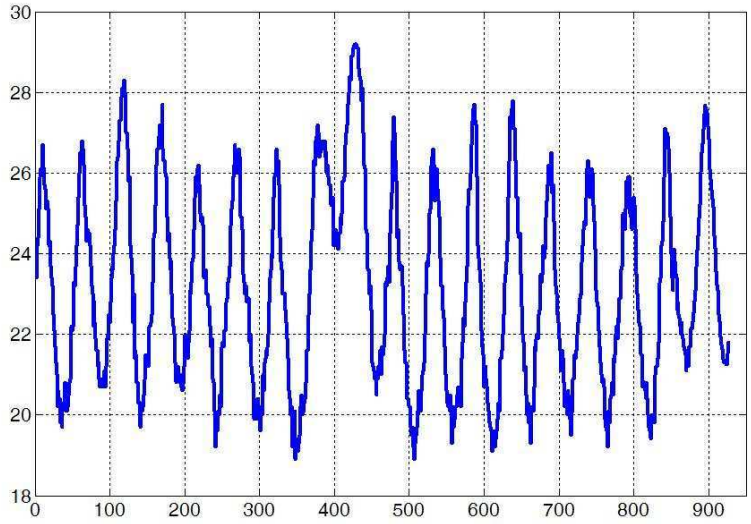

Fig. 7: Original series plus the new predicted 50 values

5 Conclusions

In this paper, a practical case of a time series prediction problem, the ESTSP'2007 proposed benchmark has been analyzed. Several paradigms and methodologies have been applied to the problem of time series prediction, however, it's very important to perform an appropriate previous analysis in order to obtain optimal prediction results. In this paper, trend and seasonality have been removed from the time series by firstly differencing and secondly removing the mean values within each season. The obtained series shows to have a very noisy behaviour and a simply AR model has been finally used to predict short term values. The desired long term prediction has been finally performed supposing a mean-valued season, and by re-seasoning and inverting the differencing process of the 50 desired values.

References

[1] Weigend, AS., Gershenfeld, NA.: Time Series Prediction: Forecasting the Future and Understanding the Past, Addison-Wesley (1993)

[2] Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994). Time Series Analysis, Forecasting and Control, 3rd ed. Prentice Hall, Englewood Clifs, NJ.

[3] L. J. Herrera, H. Pomares, I. Rojas, A. Guillén, J. González, M. Awad, A. Herrera, MultiGrid-Based Fuzzy Systems for Time Series Prediction: CATS Competition, Accepted

[4] Mills, Terence C. Time Series Techniques for Economists. Cambridge University Press, 1990.

[5] NIST/SEMATECH e-Handbook of Statistical Methods, http://www.itl.nist.gov/div898/handbook/

[6] O. Valenzuela, L.Marquez, M.Pasadas,I.Rojas, F.Rojas, A.Guillen, L.J. Herrera, Time series approximation using linear and non-linear method, Mamern'2005, May 9-11, Morocco

Identification of shock waves, model selection, and prediction

Erdal Atukeren[1]

[1] ETH Zurich, KOF- Swiss Institute for Business Cycle Research
Weinbergstrasse 35, WEH C12, CH – 8092 Zurich, Switzerland

Abstract. In this paper, we analyse the "prediction competition dataset" provided by the first European Symposium on Time Series Prediction (ESTSP). In modelling the given univariate process, we use Hamilton's (2003, J. of Econometrics) definition of a shock. The application of Hamilton's shock definition to a series with oscillatory behaviour for identifying positive and negative shock waves is a novelty of this paper. In addition, we show that it is possible to extract information on the dynamics of the shock waves and identify turning points. Furthermore, we use the generated shock wave series in the construction of a prediction model. In specifying the prediction model, we make use of Poskitt and Tremayne's (1987, Biometrika) posterior odds ratio test.

1 Introduction

The identification of the presence of individual shocks or shock waves in a given time series has important implications for the empirical modeling of the underlying process. The topic is of an interdisciplinary nature with applications ranging from solar research to financial economics. Here, we will discuss some of the methods in identifying the shocks or regime-changes in time series with a view from macroeconomics and econometrics.

There has been a time-series revolution in the economics literature in the 1980s. The main contributions, such as Nelson and Plosser [11] and Engle and Granger [6], focus on the presence and implications of non-stationarity in economic time series. For instance, if a given series is not stationary, a shock to the system has permanent effects. On the other hand, for a stationary series, a shock will have only transitory effects. In this context, Perron [13] and Rappoport and Reichlin [16] argued that macroeconomic series might indeed be stationary around a breaking trend. That is, a shock to the series reflects the effect of an exogenous event rather than being a realization from the tail of the data generating process for the series in question. Zivot and Andrews [21] further developed Perron [13] structural break test by endogenising the selection of the location of the break. Raj [15], among others, reported evidence in favour of breaking trends in the real GDP series of a number of industrialised countries.

It is also possible that a given time series contain several breaks. In other words, several shocks might disturb the process in question. Yao [19], Yin [20], and Bai and Perron [2,3,4], among others, deal with the identification of multiple breaks or shocks. The current review of the literature on multiple breaks in economic time series is provided in Jaouni and Boutahar [10].

Another strand of the literature that examines whether a given stochastic process is driven as a product of different regimes relates to non-linear time series models. Examples of such models include (self-exciting-) threshold models, (SETAR), smooth transition models (STAR), and Markov regime-switching models. The structure of these models include several segments of autoregressions with different parameter values and transition rules (threshold) across the regimes. In Markov switching models, the regime change is probabilistic and a transition probability matrix across the regimes is also a part of the model. In addition, various extensions of the generalized autoregressive conditional heteroscedasticity (GARCH) model are used for modelling the regime-switching unobserved volatility. A review of the above models and their applications in finance can be found in Tong [18] and Frances and van Dijk [7].

It is likely that the points or periods of regime-changes in non-linear time series models could be identified as breaks or periods of shocks if the structural break tests as in Perron [13], Zivot and Andrews [21], or Bai and Perron [2,3,4] were applied. The difficulty lies in the fact that one does not know the true data generating process. In this context, a pragmatic approach to what constitutes a shock without the need to identify the underlying process is provided by Hamilton [8,9]. Hamilton [8,9] argues that even a large increase or a decrease in a given time series need not constitute a shock to the series if that value is not greater (or smaller) than what was observed in the recent history of the series. However, Hamilton's definition was made in view of the oil price data. That is, economic agents' reaction function to oil price changes are also taken into account. In this sense, even a large price change would not constitute a "shock" if the economic agents (such as firms) had already seen that price level recently and adjusted their behaviour accordingly. Nevertheless, as we argue below, Hamilton's approach has further application potential in time series analysis.

In this study, we make use of Hamilton's [8,9] definition of what constitutes a "shock" to a given data series instead of employing a non-linear time series model in the analysis and forecasting of the ESTSP dataset. This is because, Hamilton's [8,9] definition of a shock does not require any particular assumptions on the nature of the data generating process.

The ESTSP competition data display highly cyclical behaviour with varying amplitudes. As such, the application of Hamilton's shock definition to data with periodic behaviour is a novelty of this paper. In the case of series with oscillations, the application of Hamilton's shock concept leads to the identification of periods of positive and negative shock waves and the calm periods in-between. In principle, it is also possible to extract information on the dynamics of the shock waves. This is especially important as it relates to the identification of turning points.

A further novelty in this paper lies in the use of the positive and negative shock waves generated by employing Hamilton's definition as explanatory variables in constructing a prediction model for the underlying series. At the model identification stage for the prediction model, we employ Schwarz's [17] Bayesian information criterion and the posterior odds ratio test developed by Poskitt and Tremayne [14].

In what follows, we first examine Hamilton's [8,9] methodology in detail. Next, we present our analysis of and the predictions for the ESTSP dataset. Some observations on the ESTSP series and the lessons from the prediction exercise conclude the paper.

2 Methodology

2.1 Hamilton's Shock Definition

As discussed in the Introduction, Hamilton [8,9] argues that any corrections, such as mean-reversion, that follow large increases or decreases should not be considered as large positive or negative changes in the variable of interest. Thus, even large increases or decreases in a given time series are not necessarily shocks to the series Hamilton's methodology of identifying what is really a shock was undertaken in the context of oil prices and the relevant history to consider is set to 12 months. In other words, a shock increase in the oil prices occurs if the current price is higher than the highest price in the last 12 months. The following equations illustrate the identification of positive and negative shocks in Hamilton's sense.

$$XP_t = X_t - \max(X_{t-1}, \dots, X_{t-n}) \text{ if } XP_t > 0; \text{ and } XP_t = 0 \text{ otherwise.} \quad (1)$$
$$XN_t = X_t - \min(X_{t-1}, \dots, X_{t-n}) \text{ if } XN_t < 0; \text{ and } XN_t = 0 \text{ otherwise.} \quad (2)$$

where XP and XN stand for the calculated positive and negative shock series, and max(.) and min(.) are the maximum and minimum functions, respectively.

2.1.1 Application of Hamilton's Methodology to the ESTSP Dataset

The ESTSP data display a highly regular cyclicality. There is, however, some variation in the amplitude and the peak-to-peak and trough-to-trough times. Nevertheless, the main irregularity is observed around 405^{th}-430^{th} observations, where the given process does not continue its downward movement but increases instead. The peak reached as a result of this disturbance leads to the highest values recorded for the process. Nevertheless, it turns out that a similar behaviour does not occur afterwards. Therefore, whatever shocked the system seems to have only transitory effects. The system, or the data-generating-process, continued its oscillation mostly in line with its pre-disturbance history.

In applying Hamilton's method to an unknown data series, the main specification issue is the determination of the relevant history window, i.e., the value of "n". In principle, one can test various alternatives and choose the best performing one (according to some statistical criteria) or a combine various alternatives. In our case, we tried N=10, N=25, and N=50 for the releveant history ($X_{t-1}, \dots, X_{t-n}$). We found that the 10-period window (i.e., N=10) tracks the dynamics of the series that leads to the peaks and trough well. The 25- and 50-period windows do not add substantial new information. Overall, the equations we employ in the calculation of the positive and negative shocks to the ESTSP series are the following.

$$PS_t = ESTSP_t - \max(ESTSP_{t-1}, \dots, ESTSP_{t-10}) \text{ if } PS_t > 0; \text{ and } PS_t = 0 \text{ otherwise.} \quad (3)$$
$$NS_t = ESTSP_t - \min(ESTSP_{t-1}, \dots, ESTSP_{t-10}) \text{ if } NS_t > 0; \text{ and } NS_t = 0 \text{ otherwise.} \quad (4)$$

where, max(.) and min (.) are the maximum and minimum functions and PS and NS stand for the positive and negative shocks, respectively. The PS and NS variables thus calculated are shown below in Figure 1.

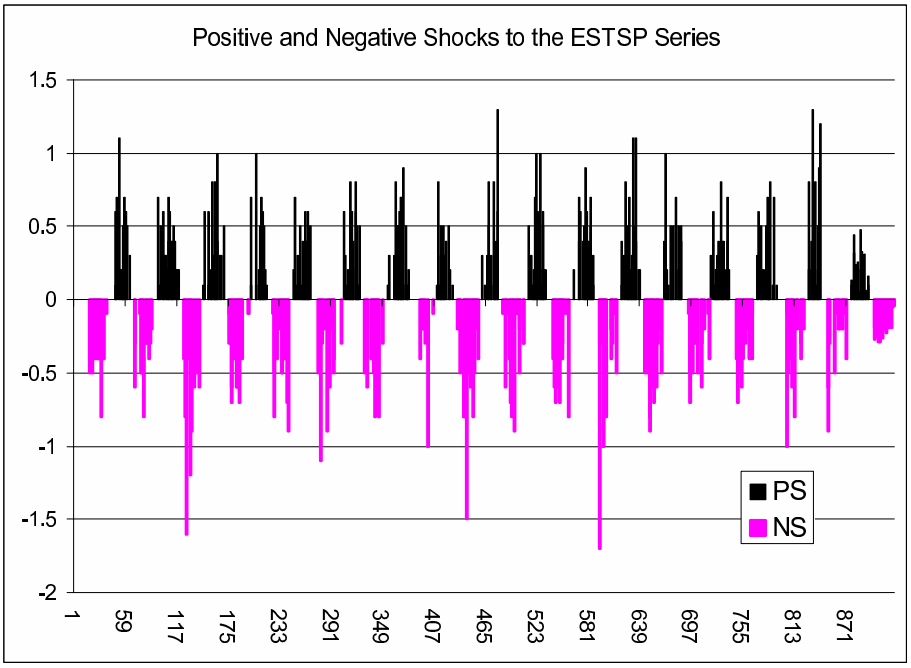

Figure 1: Positive and Negative Shocks to the ESTSP Series

The PS and NS values calculated by the application of equations (3) and (4) to the ESTSP series show that the alternating decreasing and increasing phases of the series are preserved. In addition, there are periods where there is no signal of increase or decrease (both PS and NS are equal to zero). Also, the consecutive values of the PS and NS values signal whether a positive or a negative is approaching, continuing, or coming to an end. This latter aspect is especially interesting and important in identifying the turning points and the duration of a regime.

Table 1 shows that one single drop in the value of the ESTSP series is not an enough signal for a regime change. One needs at least three consecutive non-zero values to say that the ESTSP series starts an increase or a decrease. For example, at the 277th observation, the ESTSP series is in a declining phase. This is illustrated by four consecutive negative values of NS and PS = 0. However, at the 280th-281st observations, both NS and PS are equal to zero. Does this mean that the decline has come to an end. The answer is no. PS is still zero, and indeed the 282nd observation has NS < 0. We observe a similar situation until the 289th observation, where NS < 0 and remains so for five consecutive observations. Note that PS = 0 during this time, even when NS = 0. Between the 294th and the 300th observations, both NS and PS remain at zero. This indicates that some levelling-off has been reached. The downturn has ended, but the upturn has not started yet. In such situations, there may be single drops or increases in the series. This can result from a random shock which does not disturb the series enough to trigger a new down- or up-turn. Between the 307th and the 309th observations, PS > 0, and NS = 0. This signals the upturn in the series. Despite a

break at the 310th observation (PS=0), the upturn continues into the 311th-313th observations. We have examined the complete set of PS and NS values in line with the discussion above and found that the three consecutive non-zero PS or NS values consistently signal the upturns or downturns.

Obs. No.	ESTSP	PS	NS	Obs. No.	ESTSP	PS	NS
274	25.40	0	0	**294**	20.00	0	0
275	24.90	0	-0.5	**295**	19.90	0	0
276	24.90	0	0	**296**	20.10	0	0
277	24.60	0	-0.3	**297**	20.40	0	0
278	23.50	0	-1.1	**298**	19.90	0	0
279	23.30	0	-0.2	**299**	20.20	0	0
280	23.00	0	-0.3	**300**	20.20	0	0
281	23.00	0	0	**301**	19.60	0	-0.3
282	23.20	0	0	**302**	20.20	0	0
283	22.80	0	-0.2	**303**	20.00	0	0
284	22.90	0	0	**304**	20.30	0	0
285	22.90	0	0	**305**	21.00	0.6	0
286	21.90	0	-0.9	**306**	20.60	0	0
287	21.80	0	-0.1	**307**	21.30	0.3	0
288	22.00	0	0	**308**	21.40	0.1	0
289	21.20	0	-0.6	**309**	21.60	0.2	0
290	21.00	0	-0.2	**310**	21.50	0	0
291	20.80	0	-0.2	**311**	21.70	0.1	0
292	20.30	0	-0.5	**312**	22.00	0.3	0
293	19.90	0	-0.4	**313**	22.80	0.8	0

Table 1: Signals for upturns and downturns: a numerical illustration

It is important to identify a rule for determining an upturn or a downturn and a turning point. Nevertheless, in order to make point predictions (not just directional forecasts), one also needs to take into account the amplitude of the increases and the decreases in the series. Therefore, construct a prediction model that incorporates the information provided by the positive and negative shocks and the turning point rule. In doing so, we estimate a linear regression model, where the dependent variable is the original ESTSP series and the explanatory variables are the lagged values of the ESTSP, the PS, and the NS variables. That is, the model has some autoregressive form which helps predict the level, but it is driven by the signals coming from the PS and NS variables.

2.2 Poskitt and Tremayne's Posterior Odds Ratio Test

The main question in the specification of any time series model is the selection of lags. There are many statistical criteria, such as the Akaike, Schwartz, Hannan-Quinn, etc. in selecting the dimensions of the model. In our study, we make use of the Schwarz's [17] Bayesian information criterion (SBIC). The SBIC is shown to have optimal asymptotic and small sample properties. Penm and Terrell [12] suggest the SBIC as the first choice for applied researchers. In practice, employing a statistical selection with optimal properties does not necessarily solve the model identification problem since there may still exist another model which lie within a close proximity of the chosen model.

Poskitt and Tremayne [14] argued that a posterior odds ratio test can be used to "grade the decisiveness of the evidence" in time series model identification. Poskitt and Tremayne's posterior odds ratio test uses the SBIC as the basis and questions the uniqueness of the best model (i.e., the model with the minimum SBIC value). If there are alternative models with posterior odds "close enough" to the posterior odds of the model chosen by the minimum of the SBIC, then a model portfolio – which can also be seen as a fuzzy set of models within a given proximity – can be formed. The formula for the posterior odds ratio test is

$$R = \exp[\tfrac{1}{2} T \mid SBIC_1 - SBIC_0 \mid] \qquad (5)$$

where T is the sample size and $\mid SBIC_1 - SBIC_0 \mid$ is the absolute value of the difference between the SBIC values being compared.

Then, the question is what is "close enough". Poskitt and Tremayne [5] divide the range of the R-ratio into three intervals: 1) if $R > 100$, the alternative model is unconditionally discarded; 2) if $\sqrt{10} < R < 10$, where $\sqrt{}$ is the square root function, there is "no substantial evidence" in favour of the model minimising the SBIC. That is, there exists a competing model to the specification chosen by the minimum SBIC; 3) if $1 < R \leq \sqrt{10}$, then the alternative model is a "close competitor" to the model chosen by the minimum SBIC. One should also note the fourth case, i.e., $10 < R < 100$, where the alternative model should again be discarded as non-competing. In this case, however, the best model and its close competitor are likely to be found to be statistically different by classical significance tests. A recent application of Poskitt and Tremayne's posterior odds ratio test include Atukeren [1] in the context of Granger-causality tests. Chenoweth et al. [5] also examine the R-ratio and interpret the proximity of competing models as a distance function in a Hilbert space. .In principle, Poskitt and Tremayne's posterior odds ratio and the associated model portfolio approach can be especially useful in choosing the models for making forecast combinations.

2.3 The Model

Upon testing various specifications and using the information that three consecutive non-zero values of the PS or NS variables are needed as a signal for upturn and downturn, we arrive at the following model.

ESTSP Prediction Model Dependent Variable: ESTSP Sample (adjusted): 211 - 875 Included observations: 665 after adjustments				
Variable	**Coefficient**	**Std. Error**	**t-Statistic**	**Prob.**
C	1.6660	0.3293	5.0592	5.E-07
ESTSP(-1)	0.9066	0.0630	14.395	6.E-41
ESTSP(-2)	0.0701	0.0620	1.1298	0.2590
ESTSP(-15)	-0.0265	0.0115	-2.3092	0.0212
ESTSP(-120)	-0.0231	0.0092	-2.5197	0.0120
PS(-1)	0.1017	0.1057	0.9623	0.3363
PS(-2)	0.0704	0.0788	0.8930	0.3722
PS(-3)	0.0797	0.0809	0.9855	0.3247
PS(-15)	-0.1158	0.0735	-1.5754	0.1157
PS(-25)	-0.1736	0.0745	-2.3290	0.0202
PS(-55)	0.1873	0.0798	2.3477	0.0192
PS(-200)	-0.2002	0.0758	-2.6407	0.0085
NS(-1)	-0.0731	0.1017	-0.7185	0.4727
NS(-2)	-0.0675	0.0792	-0.8517	0.3947
NS(-3)	0.1323	0.0773	1.7110	0.0876
NS(-25)	-0.1683	0.0758	-2.2191	0.0268
NS(-30)	-0.1641	0.0771	-2.1277	0.0337
NS(-76)	-0.2115	0.0761	-2.7778	0.0056
NS(-100)	0.1072	0.0721	1.4862	0.1377
NS(-200)	0.2072	0.0714	2.9039	0.0038
Adjusted R-squared	0.9774	*S.D. dependent var*		2.5029
Sum squared resid	91.324	*Schwarz criterion*		1.0480
Durbin-Watson stat	1.9998	*Prob(F-statistic)*		0.0000

Table 2: Estimation results of the Prediction Model for the ESTSP Series

Note that we include long lags extending to 200 observations back in the final equation. How meaningful is it to include such long lags in the equation? To answer this question, let us estimate the equation without the 200th lag of PS. The resulting SBIC value is 1.048969. First of all, this SBIC value is larger than the final model's SBIC value (i.e., 1.047989). Therefore, the model that does not include the 200th lag

of PS is inferior in some sense. But, how decisive is that evidence? The R-ratio between the final model and the one that excludes PS(-200) is 1.39 (despite the highly significant t-statistic). That is, the two models lie in a close proximity. Given that we do not know the frequency at which the underlying series is recorded and that the two models are closely competing, we include the PS(-200) in the model. Nevertheless, the same is not true for the PS(-199). The SBIC value with PS(-199) instead of PS(-200) is 1.058515, which yields an R-ratio of 33.11. As a result, the model with PS(-199) can be discarded as non-competing, while the term PS(-200) can be included in the model. Similar exercises can be made with the exclusion and inclusion of various lags. The procedure, however, becomes computationally intensive and some heuristic approach may be needed. In our case, we were guided by the peak-to-peak and trough-to-trough distances. In addition, we made use of a priori knowledge gained through the examination of the patterns in PS and NS variables, and included their first three values in the model in any case. Table 2 shows the estimations results.

2.4 **Predictions**

We make dynamic predictions. Our methodology involves one-step-ahead forecasts for the ESTSP dataset and the calculation of the PS and NS values at every step. The prediction for the 876th observation uses information that is completely given within the ESTSP dataset. The ESTSP prediction competition asks for the next 15 and next 50 predicted values separately. In our case, the first 15 predictions for both cases are the same. This is because the methodology employed in this paper enables the use of a single general model both for short-term and long-term predictions. Figure 2. shows the predictions

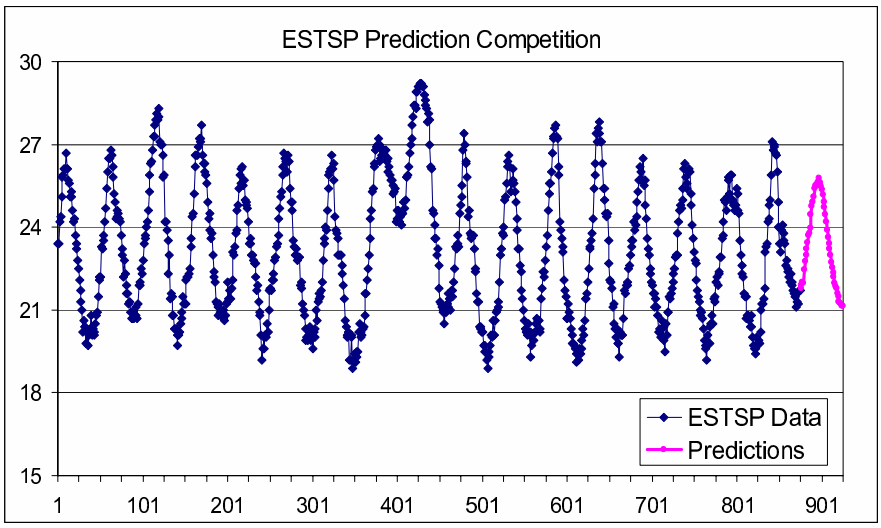

Figure 2: Original and the Predicted values of the ESTSP 2007 Dataset

In tabular form, the predicted values are shown Table 4.

Obs.	Pred.	Obs.	Pred.	Obs.	Pred.	Obs.	Pred.	Obs.	Pred.
876	21.7752	**886**	23.7698	**896**	25.6183	**906**	24.2021	**916**	21.9719
877	21.9052	**887**	23.9703	**897**	25.7815	**907**	23.9085	**917**	21.8508
878	22.0011	**888**	24.4491	**898**	25.6256	**908**	23.6186	**918**	21.7778
879	21.9735	**889**	24.7761	**899**	25.5746	**909**	23.3948	**919**	21.5869
880	22.4441	**890**	24.9299	**900**	25.4908	**910**	23.1994	**920**	21.4980
881	22.7614	**891**	25.0964	**901**	25.3822	**911**	22.9940	**921**	21.3073
882	22.9969	**892**	25.4090	**902**	25.1918	**912**	22.7318	**922**	21.2542
883	23.2119	**893**	25.4757	**903**	24.9182	**913**	22.5326	**923**	21.1998
884	23.4538	**894**	25.5178	**904**	24.7060	**914**	22.3592	**924**	21.1546
885	23.7131	**895**	25.5215	**905**	24.4516	**915**	22.1961	**925**	21.1441

Table 4: Predicted values for the ESTSP Competition

3 Conclusions

In this paper, we examined the dataset provided for the "Prediction Competition" by the "First European Symposium on Time Series Prediction" that takes place in February 2007 in Helsinki, Finland. Given the periodicity displayed in the data series, we chose the shock definition suggested by Hamilton [3] to capture the dynamics of the behaviour prior to, during, and after the expansions and contractions in the series. The given series appears to be disturbed before the middle of the sample range. However, this proves to be a transitory effect since it does not lead to a permanent path change (level, or slope) in the aftermath of the disturbance. Our approach yields insights as to the determination of the phase of the series, and the identifies a rule for the start of a new regime (increase, decrease, or a turning point). That is, for the series to enter an expansion phase, the last three values should be higher than the highest values observed in the preceding 10 values. A similar rule can be defined for the contractions as well. In making our predictions, we make use of these rules. Nevertheless, we need to make not only directional forecasts but also estimate the level of the series. This requires an analysis of the lead-lag relationship among the peaks and troughs, and brings the model identification issues in question. In that regard, we made use of Poskitt and Tremayne's posterior odds ratio test to determine whether another model is a close competitor to the model at hand or whether it can be discarded as non-competing. We end up with a final model which includes some elements of closely competing models (lags in this case). In principle, we make our predictions with a fuzzy set of competing lag structures. The final model includes short-term lags (t-1, t-2) as well as medium- (t-25, t-55), and long-term lags (t-100, t-200). As such, it is a general model that is expected to capture the data generating process. Thus, we do not develop two separate models to make short-term and long-term predictions .

The predicted values show first an increase up to the 897th observation, then a turning point, and a decrease that continues until the 925th observation. However, the process is not a uniformly increasing one in the expansion phase, nor a uniformly decreasing one in the contraction phase.

References

[1] E. Atukeren, Measuring the Strength of Cointegration and Granger-causality, *Applied Economics*, 37: 1607-1614, Taylor & Francis, 2005.

[2] J. Bai and P. Perron, Estimating and Testing Linear Models with Multiple Structural Changes, *Econometrica*, 66: 47-78, The Econometric Society, 1998.

[3] J. Bai and P. Perron, Computation and Analysis of Multiple Structural Change Models, *Journal of Applied Econometrics*, 18: 1-22, John Wiley & Sons, 2003.

[4] J. Bai and P. Perron, Critical Values for Multiple Structural Change Tests, *Econometrics Journal*, 1: 1-7, Royal Economic Society (UK), 2003.

[5] T. Chenoweth, K. Dowling,, R. Hubata and St. R. Louis, Distance and prediction error variance constraints for ARMA model portfolios, *International Journal of Forecasting*, 20: 41-52, Elsevier, 2004.

[6] R.F. Engle and C.W.J. Granger, Co-integration and Error Correction: Representation, Estimation and Testing, *Econometrica*, 55: 251-276, The Econometric Society, 1987.

[7] P.H. Frances and D. van Dijk, *Non-Linear Time Series Models in Empirical Finance*, Cambridge University Press, Cambridge, UK, 2000.

[8] J. D. Hamilton, This is What Happened to the Oil Price-Macroeconomy Relationship, *Journal of Monetary Economics*, 38: 215-220, Elsevier, 1996.

[9] J. D. Hamilton, What is an Oil Shock?, *Journal of Econometrics*, 113: 363-398, Elsevier, 2003.

[10] J. Jouni and M. Boutahar, Evidence on Structural Changes in US. Time Series, *Economic Modelling*, 22: 391-422, Elsevier, 2005.

[11] C.R. Nelson and C.I. Plosser, Trends and Random Walks in Macroeconomic Time Series: Some Evidence and Implications, *Journal of Monetary Economics*, 10: 139-162, Elsevier, 1982.

[12] J.H.W. Penm and R.D. Terrell, Multivariate subset autoregressive modelling with zero constraints for detecting 'overall causality', *Journal of Econometrics*, 24: 311-330, Elsevier, 1984.

[13] P. Perron, The Great Crash, the Oil-Price Shock, and the Unit-Root Hypothesis, *Econometrica*, 57: 1361-1401, The Econometric Society, 1989.

[14] D.S. Poskitt and A.R Tremayne, Determining a portfolio of linear time series models, *Biometrika*, 74: 125-137, Oxford University Press, 1987.

[15] B. Raj, International Evidence on Persistence in Output in the Presence of Episodic Change, *Journal of Applied Econometrics*, 6: 67-76, John Wiley & Sons, 1992.

[16] P. Rappoport, and L. Reichlin, Segmented Trends and Non-Stationary Time Series, *Economic Journal*, 99: 168-77, Royal Economic Society (UK), 1989.

[17] G. Schwarz, Estimating the dimension of a model, *Annals of Statistics*, 6: 461-464, Institute of Mathematical Statistics, 1978.

[18] H. Tong, *Non-Linear Time Series Analysis: A Dynamical System Approach*, Oxford University Press, Oxford, UK, 1990.

[19] Y.C. Yao, Estimating the Number of Change-Points via Schwarz' Criterion, *Statistics & Probability Letters*, 6: 181-189, Elsevier, 1988.

[20] Y.Q. Yin, Detection of the Number, Locations and Magnitudes of Jumps, *Communications in Statistics - Stochastic Models* , 4: 445-455, Taylor & Francis, 1988.

[21] E. Zivot and D.W.K. Andrews, Further Evidence on the Great Crash, the Oil-Price Shock, and the Unit-Root Hypothesis, *Journal of Business and Economic Statistics*, 10: 251-270, American Statistical Association, 1992.