DESIGNER NETWORKS FOR TIME SERIES PROCESSING

C. Svarer, L. K. Hansen, J. Larsen and C. E. Rasmussen
CONNECT, Electronics Institute B349
Technical University of Denmark,
DK-2800 Lyngby, Denmark
Tel: +45 4593 4207 / Fax: +45 4288 0117
emails: claus,lars,jan,ed@eiffel.ei.dth.dk

INTRODUCTION

The conventional Tapped-Delay Neural Net [14] may be analyzed using statistical methods and the results of such analysis can be applied to model optimization. In this presentation we review and extend recent efforts to demonstrate the power of this strategy within time series processing. We aim at designing compact networks using the so-called *Optimal Brain Damage* (OBD) method of Le Cun *et al.* [7]. The benefits from compact architectures are three-fold: Their generalization ability is at least comparable – mostly better, they carry less computational burden, and they are faster to adapt if the environment changes. Further, we show that the generalization error of the network may be estimated, without extensive cross-validation, using a modification of Akaike's *Final Prediction Error* (FPE) estimate [1]. The minimal FPE constitutes a useful stopping criterion for pruning [12].

TIME SERIES PROCESSING

Time series processing is an important application area for neural networks. While long time forecasts can be ruled out for chaotic systems, short time predictions may still be viable. Recent work by Priestly [11], and Weigend *et al.* [14] have established the *sunspot series* as a benchmark for time series prediction algorithms. Recently we showed how pruning by OBD can produce very compact networks for this problem [12]. We obtained networks using around one third of the parameters of the network published by Weigend *et al.* while having comparable performance. In this paper we corroborate our results on the sunspot series by application to two new fields, identification of chaotic systems (Mackey-Glass) and inverse modeling of transmission channels (channel equalization).

The basic network is characterized by a *tapped delay line* architecture with L input units, n_H hidden sigmoid units and a single linear output unit. The initial network is fully connected between layers and implements a non-linear mapping from lag space $\mathbf{x}(k) = [x(k), ..., x(k-L+1)]$, ($L$ is the length of the tapped delay line), to the real axis:

$$\widehat{y}(k) = F_{\mathbf{u}}(\mathbf{x}(k)) \qquad \widehat{y} \in \mathcal{R}, \tag{1}$$

where $\mathbf{u} = [\mathbf{w}, \mathbf{W}]$ is the N-dimensional weight vector and $\widehat{y}(k)$ is the prediction of the target signal $y(k)$. The non-linear mapping can be written as:

$$F_{\mathbf{u}}(\mathbf{x}(k)) = \sum_{j=1}^{n_H} W_j \tanh\left(\sum_{i=0}^{L-1} w_{ij} x(k-i) + w_{i0}\right) + W_0, \tag{2}$$

where n_H is the number of hidden units. W_j are the hidden-to-output weights while w_{ij} connect the input and hidden units.

TRAINING

The objective of the training procedure is model identification. Hence, the network weights, $\mathbf{u}$, are trained to recognize the short time structure of the time series. We use the sum of squared errors to measure the performance of the current network:

$$E_{\text{train}} = \frac{1}{p} \sum_{k=1}^{p} \left[y(k) - F_{\mathbf{u}}(\mathbf{x}(k))\right]^2, \tag{3}$$

where p is the number of training examples.

A simulator has been developed based on *batch mode*, second order local optimization. A direct matrix-inversion method is used for identification of the hidden-to-output weights [2], while a pseudo Gauss-Newton method (diagonal approximation) is used for identification of input-to-hidden weights, see e.g. [5]. To ensure numerical stability and for assisting the pruning procedure we augment the cost-function with a weight decay term. The cost-function can then be written as:

$$E = E_{\text{train}} + \frac{\alpha_w}{p} \sum_{ij}^{N_w} w_{ij}^2 + \frac{\alpha_W}{p} \sum_{j}^{N_W} W_j^2, \tag{4}$$

where N_w, N_W are the numbers of weights and thresholds in hidden and output units, respectively. Further, α_w, α_W is the weight decay parameters of the hidden and output layers, respectively.

The second order pseudo Gauss-Newton method used for identification of input-to-hidden weights can be written as:

$$\Delta w_{ij} = -\eta \left(\frac{\partial E_{\text{train}}}{\partial w_{ij}} + \frac{2\alpha_w}{p} w_{ij}\right) \Bigg/ \left(\frac{\partial^2 E_{\text{train}}}{\partial w_{ij}^2} + \frac{2\alpha_w}{p}\right) \tag{5}$$

where the gain parameter η is used to secure that all weight updates lead to a decrease in the cost-function. Before each step η is initialized to 1 and iteratively diminished by powers of two until the step leads to a decrease in the cost-function. As e.g. in [7] we approximate the second derivative by the positive semi-definite expression:

$$\frac{\partial^2 E_{\text{train}}}{\partial w_{ij}^2} \approx \frac{2}{p} \sum_{k=1}^{p} \left(\frac{\partial F_{\mathbf{u}}(\mathbf{x}(k))}{\partial w_{ij}} \right)^2 . \tag{6}$$

PRUNING BY OPTIMAL BRAIN DAMAGE

The OBD method proposed by Le Cun *et al.* [7] was successfully applied to reduce large networks for recognition of handwritten digits [8]. The basic idea is to estimate the increase in the *training error* when deleting weights. The estimate is formulated in terms of weight *saliencies* s_l:

$$\delta E_{\text{train}} = \sum_{l \in D} s_l \equiv \sum_{l \in D} \left(\frac{2\alpha}{p} + \frac{1}{2} \frac{\partial^2 E_{\text{train}}}{\partial u_l^2} \right) u_l^2 , \tag{7}$$

where u_l is a component of $\mathbf{u}$ and the sum runs over the set of deleted weights D. The saliency definition used here takes into account that the weight decay terms force the weights to depart from the minimum of the training set error.

The major assumptions entering the derivation of OBD are: 1) Terms of third and higher orders in the deleted weights can be neglected. 2) The off-diagonal terms in the Hessian, $\partial^2 E_{\text{train}} / \partial u_l \partial u_{l'}$, can be neglected. Computationally, the second order (diagonal) terms are reused from the training scheme eq. (6). We refrain from operations involving the full Hessian, which scales poorly for large networks. The recipe allows for *ranking* the weights according to saliency. The question of how many weights it may be possible to delete was answered in [12]. We applied Akaike's FPE estimate [1, 9] of the test error in terms of the training error. In its standard form it reads:

$$\widehat{E}_{\text{test}} = \frac{p+N}{p-N} E_{\text{train}} , \tag{8}$$

where p is the number of training samples, and N is the number of parameters in the model. The left hand side of eq. (8) is the average generalization error, averaged over all possible training sets of size p. The estimate is based on linearization of the networks as regards the fluctuations in the weights resulting from different training sets. For a discussion of the approximations entering this estimate see [6]. The relation expresses the fact that the training and test errors are biased estimates of the noise level because each parameter during training has "absorbed" noise from the training samples.

Since we have regularized the training procedure by weight-decay terms α_w, α_W, hence, suppressed the ability of the (otherwise) ill-determined parameters to model noise, we need to modify the standard FPE estimate by replacing the total number of parameters with the *effective* number of parameters see e.g. [6, 10, 12]:

$$\widehat{E}_{\text{test}} = \frac{p + N_{\text{eff}}}{p - N_{\text{eff}}} E_{\text{train}}, \tag{9}$$

$$N_{\text{eff}} = \sum_{ij}^{N_w} \left(\frac{\lambda_{ij}}{\lambda_{ij} + 2\alpha_w/p} \right)^2 + \sum_{j}^{N_W} \left(\frac{\Lambda_j}{\Lambda_j + 2\alpha_W/p} \right)^2 . \tag{10}$$

where the λ's are the second derivatives already computed in eq. (6), $\lambda_{ij} \equiv \partial^2 E_{\text{train}} / \partial w_{ij}^2$, $\Lambda_j \equiv \partial^2 E_{\text{train}} / \partial W_j^2$.

With the above tool we can obtain a generalization error estimate for each pruned network. By selecting the network with the lowest estimated generalization error we have developed the stop criterion sought.

EXPERIMENTS

In [12] we tested the system on the classical sunspot series. In this paper the methods are tested on two other typical signal processing problems. The first is a standard problem of nonlinear dynamics viz. the Mackey-Glass chaotic time series, and the second concerns channel equalization. For both systems errors are computed as

$$E_{\text{set}} = \frac{1}{\sigma_{\text{total}}^2 \cdot p_{\text{set}}} \sum_{k=1}^{p_{\text{set}}} \left[y(k) - F_{\mathbf{u}}(\mathbf{x}(k)) \right]^2 , \tag{11}$$

where p_{set} is the number of examples in the data (train or test) set in question, and σ_{total}^2 is the total variance of $y(k)$ on the training and test set.

Mackey-Glass Chaotic Dynamics

The Mackey-Glass attractor is a non-linear chaotic system described by the following equation:

$$\frac{dz(t)}{dt} = -bz(t) + a \frac{z(t-\tau)}{1 + z(t-\tau)^{10}} \tag{12}$$

where the constants are $a = 0.2$, $b = 0.1$ and $\tau = 17$. The series is resampled with sampling period 1 according to standard practice.

We aim at identifying the underlying dynamic model, from this chaotic time series. The network configuration is $L = 16$, $n_H = 10$ and we train to implement a six step ahead prediction. That is, $\mathbf{x}(k) = [z(k-6), z(k-12), \cdots, z(k-6L)]$ and $y(k) = z(k)$.

In Fig. 1 the normalized training and test errors cf. eq. (11), and the FPE error are sketched for a training set size of 500 and 1000 examples, the test set comprises 8500 examples. Weight decays were set to $\alpha_w = 0.001$, $\alpha_W = 0.001$. With higher weight decays we were unable to train the networks to error levels like those reported. On the other hand, with smaller weight decays the second order optimization scheme is plagued by numerical problems, also leading to higher errors and a lower yield of useful architectures. It is seen that the stop criterion is able to select the optimal network for both training set sizes. As expected, a small training set can only "justify" a network with few parameters.

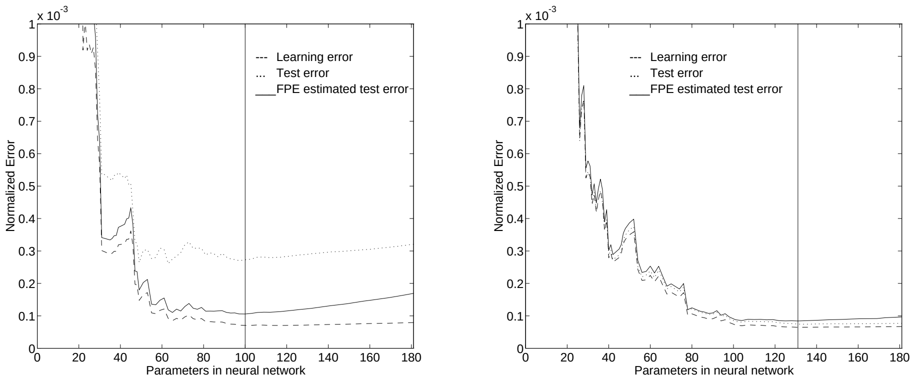

Figure 1: The evolution of training and test errors during pruning for the Mackey-Glass time series (left) training set size 500, (right) training set size 1000. The FPE estimate of the test error is based on eq. (9). The vertical line indicates the network for which the *estimated* test error is minimal. The figures show that a small training set will favor a simple model with few parameters.

In Fig. 2 we show the attractors for the original Mackey-Glass series (a), and three network architectures whose outputs are fed back as new inputs, thus performing iterative predictions. Panel (b) displays the pruned network. In (c) the attractor for the fully connected network trained on 1000 examples is shown, and finally the attractor of the pruned network trained on 1000 examples is shown in (d).

In Table 1 we compare the performances of pruned networks with those of fully connected nets, a linear model, and with a K-nearest-neighbor linear model [4]. It is interesting to note that the performance of the networks is similar to the nearest neighbor estimate. The latter involves finding the neighbors among the 1000 training examples and computing a regularized linear estimate of the output. The model has been optimized by leave-one-out cross-validation and the computational overhead (time and storage) is significant for this model. Also it is noteworthy that the pruned network, retrained without weight decay, is clearly superior. We have found that weight decay greatly assist pruning and optimization for the networks with superfluous resources; however, for the optimized pruned architectures we remove the weight decay and retrain to fine tune the performance [12].

In each case we compute the correlation with the true Mackey-Glass at-

Normalized error (mean square) on Mackey-Glass time series

	Train	**Test**	**No. of par.**
5-nearest-neighbors	4.80×10^{-7}	8.37×10^{-5}	–
Linear model[1]	9.70×10^{-2}	9.58×10^{-2}	17
Fully connected network[2]	6.30×10^{-5}	7.67×10^{-5}	181
Pruned network[3]	6.45×10^{-5}	7.45×10^{-5}	131
Pruned network[4]	3.12×10^{-5}	3.68×10^{-5}	131

Table 1: 1) Linear model is a single linear unit. 2) The initial pre-pruned networks, trained with the same weight decay terms as used during pruning. 3) Pruned network. 4) Pruned networks retrained *without* weight decay.

tractor, as described in [15]. The attractor is sampled on a grid of 20×20 points for the correlation estimate. The correlation coefficients for two independent samples of 8850 points of the true attractor is 0.991. For the fully connected network trained on 1000 examples it is 0.922, for the pruned network trained on 1000 examples 0.948. In order to compare with [15] we also completed a pruning session based on on 500 examples, and for the pruned network we found a correlation coefficient of 0.934 (with weight decay), while the coefficient for a feed-forward network with second order couplings were found to be 0.917 in [15]. For networks trained on 500 examples only, it turns out that, retraining without weight decay deteriorates the performance, indicating that pruning has not identified the optimal architecture.

We conclude that the Designer Networks more accurately identify the underlying dynamics, than straight fully connected networks.

Inverse Modeling: Channel Equalization

We have also implemented the toy-model studied by Day *et al.* [3], involving a non-linear channel sandwiched between low-pass filters with identical transfer functions $H_1(z) = 0.2602 - 0.9298z^{-1} + 0.2602z^{-2}$ and the non-linearity of the channel is given by the simple map $v = 5u|u|(1-u^2)\,/(1+u^2)$. The driving signal is a Gaussian i.i.d. sequence (unit variance, zero mean) filtered through a third-order *Butterworth* low-pass filter with the transfer function

$$H_2(z) = \frac{0.727 + 2.18z^{-1} + 2.18z^{-2} + 0.727z^{-3}}{1.000 - 2.327z^{-1} + 2.072z^{-2} - 0.686z^{-3}} \cdot 10^{-3}, \qquad (13)$$

The network is trained, using weight decays $\alpha_w = 0.01$, $\alpha_W = 0.01$, to invert the channel and reconstruct the input as illustrated in Fig. 3. As input to the *network* we use a tapped delay line with 25 taps (noisy outputs from the the channel). The signal to be predicted is delayed 15 samples in order to ensure causal filtering. The training set comprises 2000 examples, which is a minute fraction of the training set used in [3]. In the left panel of Fig. 4 the

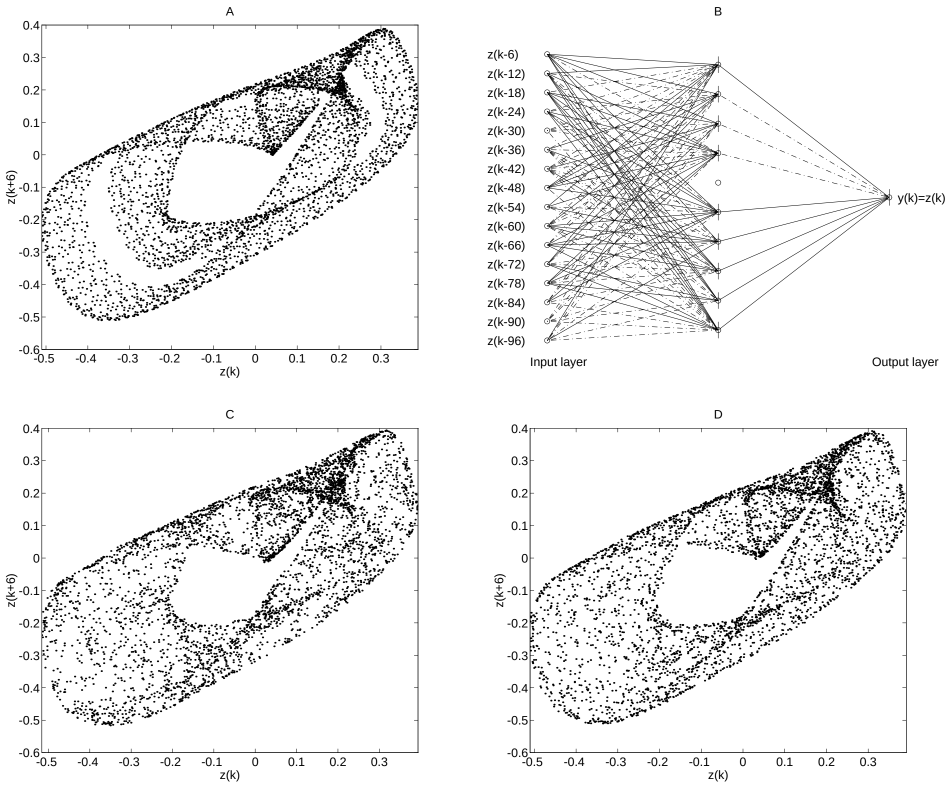

Figure 2: Phase-state plot ($z(k)$ versus $z(k+6)$) of a) The Mackey-Glass attractor. The correlation coefficient for two independent samples of 8850 points is 0.991. b) The pruned network obtained with a training set of 1000 examples. c) The estimated attractor for fully connected network trained on 1000 examples, having 181 weights and thresholds and a correlation coefficient of 0.922. d) The estimated attractor for the pruned network trained on 1000 examples (retrained without weight decay) having 131 weights and a correlation coefficient of 0.956.

evolution of the training and test errors during a pruning session are shown, and further the FPE estimate of the test error. We note that the minima of the estimated test error and the experimental test error (2000 samples) are compatible, hence confirming that the statistical approach is viable. In the right panel of Fig. 4 we show the pruned network obtained. As expected we see that the network make use of the inputs which in lag space are closest (in time) to the output to be estimated.

We present our test errors relative to the performance of a linear model. With a linear model with input as the networks obtains a normalized mean-square error defined in eq. (11) of 0.130. For the network the test error is reduced to 0.039. This is a reduction of the test error with a factor 0.3. In

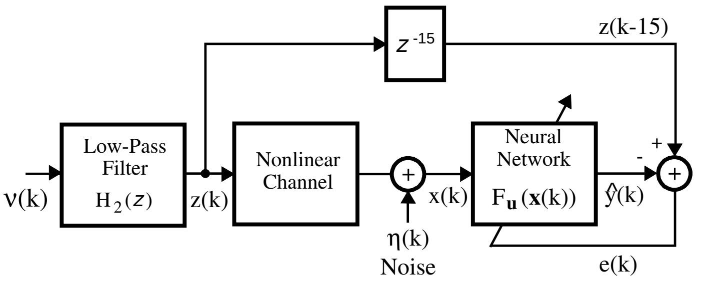

Figure 3: Channel equalization. The non-linear channel is driven by a low-pass filtered white noise signal $z(k)$. Output from the channel is denoted $x(k)$. The neural network is trained to reproduce the channel input $z(k)$ based on a sample of the noisy output $x(k)$. The variance of the channel Gaussian i.i.d. noise $\eta(k)$ is 30dB below the variance of the channel output.

[3] the reduction was from 0.0625 to 0.0256 for the back-prop net[1], which amounts to a reduction by a factor of 0.41.

CONCLUSION

We have corroborated our earlier results for the sunspot series [12] by new results for chaotic time series prediction and channel equalization. We suggested to optimize network architectures using the Optimal Brain Damage pruning scheme combined with our new statistical stopping criterion.

For the Mackey-Glass time series the pruned networks used almost all delayed input signals, while the networks for the channel equalizer mainly used the delayed inputs which are closest in time to the output signal. This shows that the pruning scheme effectively captures the relevant connections and removes unimportant hidden units.

We succeeded in designing compact neural networks which generalized better than the unpruned networks. A significant gain in generalization abil-

[1] The errors in [3] are defined as the normalized root-mean-square error. Furthermore, it should be noted that the error for the linear network on our data is significantly higher than the error of [3]. This is not due to the different training set sizes used, since the linear model of [3] converged very quickly. Rather, the non-linearity in our data set is more noticeable. It may therefore be more difficult to invert the channel with the present data set.

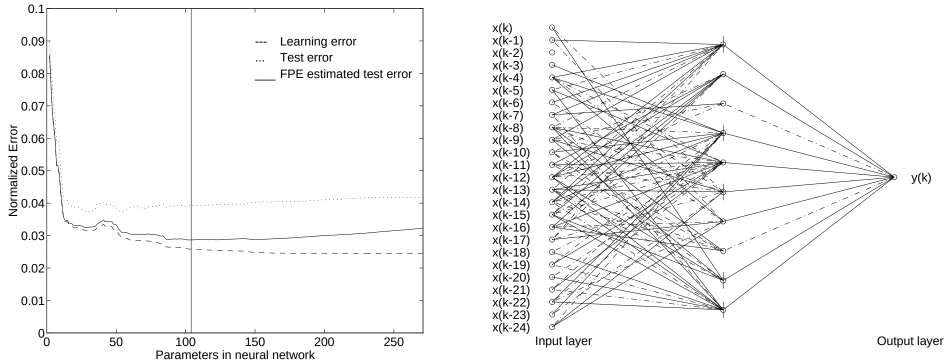

Figure 4: The left panel shows the evolution of training and test errors during pruning for channel equalization. The FPE estimate of the test error is based on eq. (9). The vertical line indicates the network for which the *estimated* test error is minimal. In the right panel we show the resulting, pruned, network. A vertical bar through a unit indicates an active threshold, full lines indicate positive connections, dash-dotted lines negative connections.

ity is achieved by retraining the pruned network without weight decay. However, note that weight decay is essential for the success of the pruning. Hence, we recommend pruning by Optimal Brain Damage supplemented by our newly developed stopping criterion as a tool for adapting "Designer Networks" for time series processing.

ACKNOWLEDGMENTS

This research was supported by the Danish Natural Science and Technical Research Councils through the Computational Neural Network Center (CONNECT).

REFERENCES

[1] H. Akaike: "Fitting Autoregressive Models for Prediction". Ann. Inst. Stat. Mat. vol. 21, 243-247, 1969.

[2] S.A. Barton: "A Matrix Method for Optimization a Neural Network". Neural Computation, vol. 3, 450-459, 1990.

[3] A.P. Day, M.R. Davenport, and D.S. Camporese: "Dispersive Networks for Nonlinear Adaptive Filters". In Neural Networks For Signal Processing II; Proceedings of the 1992 IEEE-SP Workshop, (Eds. S.Y. Kung, F. Fallside, J.Aa. Sørensen, and C.A. Kamm), IEEE Service Center, 464-473, 1992.

[4] J.D. Farmer, and J.J. Sidorowich: "Exploiting Chaos to Predict the Future and Reduce Noise", Technical Report LA-UR-88, Los Alamos National Laboratory, 1988.

[5] J. Hertz, A. Krogh and R.G. Palmer: Introduction to the Theory of Neural Computation, Addison Wesley, New York, 1991.

[6] J. Larsen: Design of Neural Network Filters. Ph. D. Thesis, Electronics Institute, Technical University of Denmark, March 1993.

[7] Y. Le Cun, J.S. Denker, and S.A. Solla: "Optimal Brain Damage". In Advances in Neural Information Processing Systems 2, 598-605, Morgan Kaufman, 1990.

[8] Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jakel: "Handwritten Digit Recognition with a Back-Propagation Network". In Advances in Neural Information Processing Systems 2, 396-404. Morgan Kaufman, 1990.

[9] L. Ljung: System Identification: Theory for the user, Prentice-Hall, Information and System Sciences series, 1987.

[10] J.E. Moody: "Note on Generalization, Regularization and Architecture Selection in Nonlinear Systems". In Neural Networks For Signal Processing; Proceedings of the 1991 IEEE-SP Workshop, (Eds. B.H. Juang, S.Y. Kung, and C. Kamm), IEEE Service Center, 1-10, 1991.

[11] M.B. Priestly: Non-linear and Non-stationary Times Series Analysis, Academic Press, 1988.

[12] C. Svarer, L.K. Hansen, and J. Larsen: "On Design and Evaluation of Tapped Delay Line Networks", In Proceedings of the 1993 IEEE International Conference on Neural Networks, San Francisco, vol. 1, 46-51, 1993.

[13] H. Tong and K. S. Lim: "Threshold autoregression, limit cycles and cyclical data". Journ. Roy. Stat. Soc. B, vol. 42, 245, 1980.

[14] A.S. Weigend, B.A. Huberman, and D.E. Rumelhart: "Prediction the future: A Connectionist Approach", Int. J. of Neural Systems, vol. 1, 193-209, 1990.

[15] N.H. Wulff and J.A. Hertz: "Prediction with Recurrent Networks". In Neural Networks For Signal Processing II; Proceedings of the 1992 IEEE-SP Workshop, (Eds. S.Y. Kung, F. Fallside, J.Aa. Sørensen, and C.A. Kamm), IEEE Service Center, 464-473, 1992.