PCN '95, Istanbul, Turkey

On the prediction of the stochastic behavior of time series by use of Neural Networks - performance analysis and results

M. D. Eberspaecher
University of Stuttgart, Institute of Communications Switching and Data Technics
Seidenstrasse 36, 70174 Stuttgart, Germany, Phone: +49 711 1212482, Fax: +49 711 1212477, E-mail: eberspaecher@ind.uni-stuttgart.de

Abstract

In time series theory, the prediction of future values is a widely discussed subject. There are manyfold methods to derive models from data. One of the main objectives is to obtain the model parameters. Some proposals use self adapting techniques like Neural Networks to estimate the model parameters. Most of these approaches predict one future value of a time series. Some simulation tasks require models for traffic sources that are closely related to time series prediction though there exist different requirements. One of them is that a simulated traffic source should show the same stochastic behavior as a reference source. In this paper a procedure is presented that automatically adapts to a given reference source in the sense described above.

Keywords

Time series, prediction, Neural Networks, source modelling

1 INTRODUCTION

General

The analysis of time series is an extensively developed area of mathematics. There are many approaches to model dynamic systems. They can be classified as follows: linear, linear stochastic, nonlinear and nonlinear stochastic dynamic systems. Nonlinear systems may show chaotic behavior, depending on system parameters. There is a sliding transition from systems with a random disturbance to systems with probabilistic transitions between states. In addition seasonal effects and trends may be observed.

Depending on the underlying system an appropriate model has to be found. ARMA and ARIMA models (Auto-Regressive-Moving-Average and Auto-Regressive-Integrated-Moving-Average, respectively) represent simple methods to model linear and linear stochastic systems in a suitable way. ARIMA models often are used when trends have to be considered. For other

sequences, more sophisticated methods are required (Janacek, 1993, Harvey, 1993, Hamilton, 1994).

Nonlinear models are necessary to adequately model nonlinear and nonlinear stochastic systems. Here chaotic behavior of time series might occur and has to be identified since chaotic series must be treated differently than stochastic time series, see Scargle (1992).

However, many problems need their own specific solutions (Brillinger, 1992, Weigend, 1993). Most of these approaches share the following characteristics: firstly, as much information as possible about the characteristics of the underlying data is collected and, secondly, a model that covers the essential features is deduced.

Most procedures that deal with forecasting future values predict one (the next) value based on a set of N past values. The selection of N is not trivial since it determines the prediction quality and depends on the observed dynamic system. Some remarks how to determine useful values of N can be found in Scargle (1992).

All methods mentioned above share one disadvantage: they are inflexible in terms of changing stochastic behavior of the underlying data. These changes have to be taken into account by the model and increase its complexity very much.

Time series models that are able to deal with changing parameters should be based on an architecture that is inherently able to automatically adapt to these changes. This architecture could be based on a Neural Network. Until today there are not many approaches that use a Neural Network (NN) architecture. Their advantage primarily consists in their ability to learn a given behavior without the exact knowledge of the underlying system and without difficult analysis needed for modelling. One disadvantage is that no detailed and understandable model of the underlying system is built.

These models are most often used for prediction: Chakraborty (1992) presents a neural network to multivariate time series analysis, Deppisch (1994), Lowe (1994) and Hudson (1994) use neural network algorithms to predict chaotic time series. Mozer (1993) presents a general taxonomy of neural net architectures for processing time-varying patterns. In Tang (1994) a neural net approach is compared to the Box-Jenkins methodology. Wan (1993) presents a somewhat different method that uses a neural net with internal delay lines, i. e. a neural net with inherent memory.

Source modelling is an area where the generation of future values is frequently used.

Source modelling

In source modelling the generation of deterministic new values from given data such as forecasting exchange rates is often of no particular interest. In contrast to that, a random data sequence is generated by a stochastic model. Every new value is randomly chosen from a given distribution depending on the state of the model. State changes are most often defined by a state transition probability function that may be nonlinear and depend on past states. Source modelling is frequently used for traffic generation in the simulation of communication networks, in the simulation of manufacturing plants or in measurement technology. Since this paper concentrates on communications all the examples will relate to this area.

Multiple traffic sources built from the same model must be statistically independent from each other. A simple reproduction of measured data from a file (play back) is not sufficient. A shifted play back from a file where one traffic sequence starts at one point and another traffic sequence starts at another point of the file is not sufficient, too, because of the strong correlations (especially when the file is short or when many sources are needed). Even in the case of very long files problems might occur in large simulations. This leads to the conclusion that reproduction from files is inflexible.

Like in conventional time series analysis ARMA models can be used for source modelling but they don't fit very well because of nonlinearities in almost all systems. There are some approaches employing Neural Networks that avoid this problem but most of them don't fulfill all requirements for source modelling.

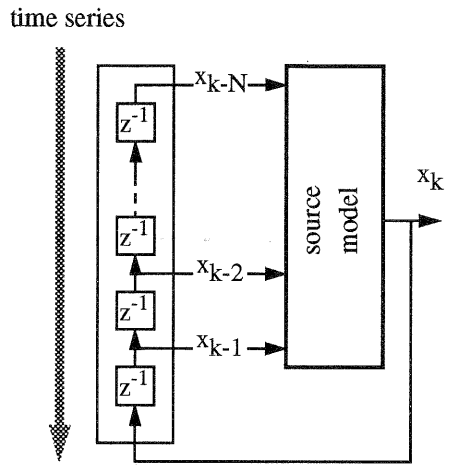


Figure 1 Principle of source model.

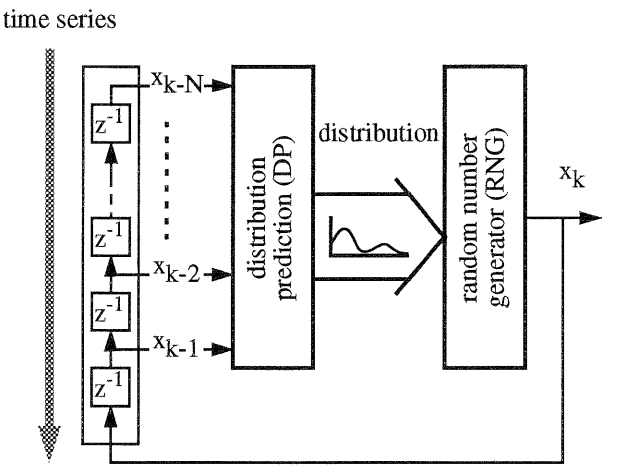


Figure 2 Extended principle of source model.

In Tarraf (1993, 1994) NNs are used for modelling an ATM cell stream (Asynchronous Transfer Mode) without adding a random component. This kind of modelling is sufficiently exact, but not very useful for simulation.

How can traffic models be obtained? This is a simple task when only distributions are of interest and no correlations. Distributions might stem from measurement. The task can become very difficult when correlations must be taken into account. In this case an ARMA process might be profitably used, but how to obtain the model parameters? Another problem is that ARMA models are normally driven by white noise with the consequence that negative values are possible in any case, even if they are not allowed. Gruenenfelder (1991) shows an example for source modelling using an ARMA model. Here the parameters are partly estimated from the measured data and partly calculated via the frequency domain.

Most of the cited approaches (Gruenenfelder (1991) is an exception) lead to a deterministic behavior of the model in the way that they calculate a new value based on some observations, without adding any noise. Neural Network based models that adapt to a given time series during a learning phase therefore do not learn the stochastic behavior but the conditional expectation of future values.

In the following parts of this paper a new method is presented that adapts a source model to many different random processes or time series. It uses Neural Networks to automatically learn the stochastic behavior of the underlying data and therefore avoids some of the problems of other models. The adaptation process is fully automated and only a few topological model parameters have to be estimated from the data.

Automatic source identification

The advantage of automatic source identification is a gain in productivity and saving of money since computational power is much cheaper than man power today. The disadvantage is the reduced possibility of interpretation of the generated source model. Only very few topological parameters have to be estimated from the data. All random processes that are weakly stationary and that have some seasonal effects can be modelled.

The objective of the identification procedure is to model the distribution and autocorrelation of a time series as good as possible. Some simple tools for evaluation are presented later in this paper, see section 3.

Figure 1 shows one principle of this approach. Suppose that the content of the box named "source model" is already adapted to the data. The scalar value x_k is the output value of the modelled traffic source, N time delayed output values form the input vector $\boldsymbol{i}_k$

$$\boldsymbol{i}_k = (x_{k-1}, x_{k-2}, \ldots, x_{k-N}) . \tag{1}$$

The elements of $\boldsymbol{i}_k$ form the embedding space coordinates with dimension N. To determine a suitable value of N a simple approach is used. The sample autocorrelation function of the observed time series is calculated and evaluated. In the case of periodic signals N is chosen to be greater than the period length. In the case of vanishing autocorrelation N is chosen to be equal the lag where the absolute value of the autocorrelation falls below $1.96/\sqrt{T}$, where T is the sample size (see Figure 8,a). It is assumed that autocorrelation values less than this value are based on white noise, see Harvey (1993).

The shift register forms a memory of the past. This is the only memory of the model. The vector $\boldsymbol{i}_k$ is fed into the "source model," the output of which is the newly generated value. This output is fed back to the shift register, delayed by one step. So the number generating loop is closed.

Figure 2 provides a more detailed view. The source model is now divided into two parts, the distribution prediction (DP) and the random number generation (RNG). The RNG simply draws a new number according to the distribution density at its input. The general inverse-transform method is used as RNG algorithm, see Law (1991). For every input vector the DP block computes (predicts) the distribution of the following output value.

This model can be described by some equations. The internal state of the DP is a function of the input vector $\boldsymbol{i}_k$:

$$S_k = h(\boldsymbol{i}_k) = h(x_{k-1}, x_{k-2}, \ldots, x_{k-N}) \; . \tag{2}$$

The density function at the output can be expressed as a function of the internal state and therefore depends on the input vector $\boldsymbol{i}_k$, too:

$$G_k = g(S_k) = g(h(x_{k-1}, x_{k-2}, \ldots, x_{k-N})) \; . \tag{3}$$

The output value of the model, x_k, is chosen according to the density G_k.

In Section 2 the components of the distribution prediction including the learning process are described. In section 3 some measures for performance evaluation are introduced and in section 4 some examples of time series that are learned and predicted by applying the new source model are presented.

2 DISTRIBUTION PREDICTION

At first the scenario shown in Figure 3 is examined. N values of a time series preceding the current value x_k are fed to a black box named "distribution prediction." Inside this box the prediction of the distribution of the actual value is computed. For each vector $\boldsymbol{i}_k$ at the input there is a distinct distribution at the output.

Before continuing the theoretical model description the correspondence between distribution and correlation has to be clarified. The occurrences of vectors $\boldsymbol{i}_k$ obey an N-dimensional distribution $\hat{g}$ in the embedding space $\Re^N$. From $\hat{g}$ the autocorrelation of the series x_k can be derived, see Papoulis (1984). Therefore it is sufficient for the model to learn a good approximation of $\hat{g}$ to model the autocorrelation of the underlying time series.

The actual value x_k and the input vector $\boldsymbol{i}_k$ are now treated as one $(N+1)$-dimensional vector $\boldsymbol{x}_k = (x_k, \boldsymbol{i}_k)$ with an $(N+1)$-dimensional distribution function

$$F(x_k, x_{k-1}, \ldots, x_{k-N}) = P\{X_k \le x_k, \ldots, X_{k-N} \le x_{k-N}\} \tag{4}$$

and density function

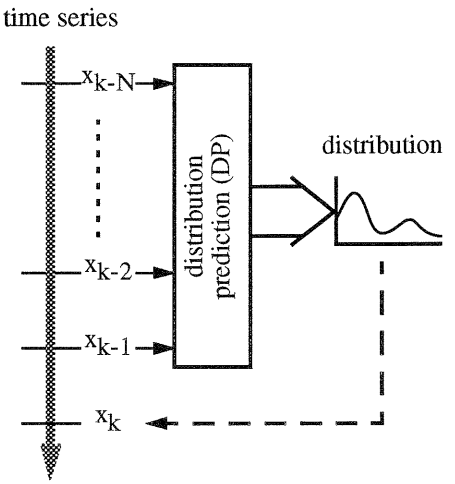


Figure 3 Principle of distribution prediction.

$$f(x_k, x_{k-1}, \ldots, x_{k-N}) = \frac{\partial^N}{\partial x_k \ldots \partial x_{k-N}} F(x_k, \ldots, x_{k-N}). \tag{5}$$

Without loss of generality k is now set to 0 for simplicity. Then the density function becomes

$$f(x_0, x_{-1}, \ldots, x_{-N}). \tag{6}$$

For distribution prediction the conditional distribution of x_0 for distinct values of the input vector $\boldsymbol{i}_k$ is needed. This is done by local approximations of parts of this distribution. For the local approximation the N-dimensional embedding space that belongs to the input vector is quantized into M discrete vectors

$$\hat{\boldsymbol{p}}_i = (\hat{x}^i_{-1}, \ldots, \hat{x}^i_{-N}), \qquad i = 1, \ldots, M. \tag{7}$$

This task is carried out by a vector quantizer (VQ).

Each vector $\hat{\boldsymbol{p}}_i$ points to the center of a region I_i of $\Re^N$. In these regions the density function of x_0, (6), is approximated by the function $\hat{f}_i(x_0)$. The union of all regions I_i forms the N-dimensional space $\Re^N$.

The approximation in region I_i is defined by the mean value of density (6) in this region:

$$\hat{f}_i(x_0) = \frac{1}{Vol(I_i)} \cdot \int_{I_i} \ldots \int f(x_0, t_1, \ldots, t_N) dt_1 \ldots dt_N \tag{8}$$

where

$$Vol(I_i) = \int_{I_i} \ldots \int dt_1 \ldots dt_N . \tag{9}$$

This leads to the following error inside region I_i (the error measure is the squared difference between $f(\ldots)$ and $\hat{f}_i(\ldots)$):

$$E_i = \int_{-\infty}^{\infty} \left[\int_{I_i} \ldots \int f^2(x_0, t_1, \ldots, t_N) dt_1 \ldots dt_N \right] dx_0 - Vol(I_i) \cdot \int_{-\infty}^{\infty} \hat{f}_i^{\,2}(x_0) dx_0 \tag{10}$$

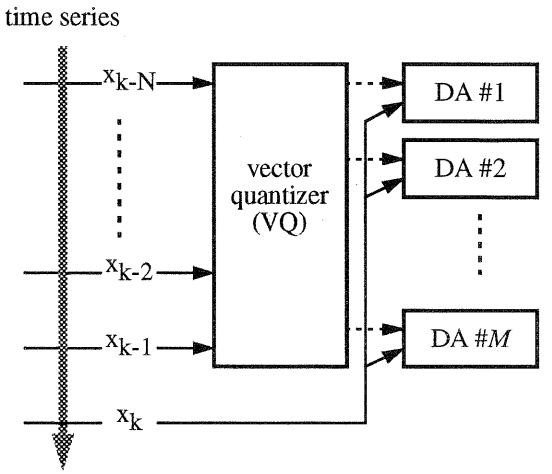


Figure 4 Learning of DP.

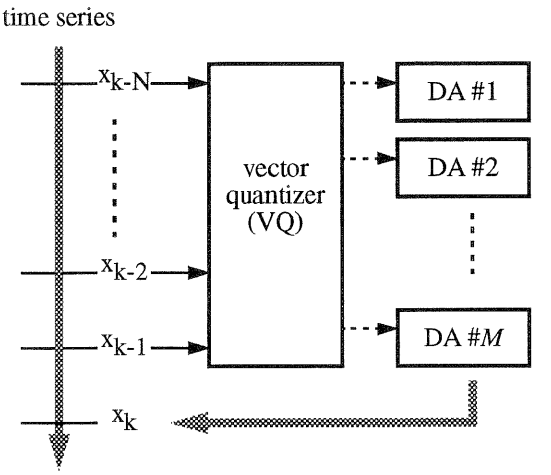


Figure 5 Prediction with VQ and DA's.

To compute the total error that results from quantization some features of the vector quantizer need to be known. This VQ adapts to the x_k in a way that all regions I_i are equally probable for the given time series. The algorithm used is partly taken from literature and is described in detail in Appendix A. The resulting total error is:

$$E_G = \sum_{i=1}^{M} E_i \tag{11}$$

E_G decreases for an increasing number M of regions.

The approximated density $\hat{f}_i(x_0)$ is obtained from the given time series, too. A new Neural Network algorithm was investigated that is able to form a distribution when a sequence of data is offered to it. This algorithm is presented in the next subsection.

Figures 4 and 5 show the relationship between vector quantizer and distribution approximation (DA). Whenever an input vector falls into a region of the VQ, the corresponding DA is chosen for learning or prediction. In other words, the VQ is responsible to detect all correlations, the DA's are responsible for representing distributions.

Figure 4 shows the learning case. The actual value x_k is needed here as input for the actual DA to adapt the distribution. In the case of prediction (Figure 5), when learning is completed, one DA is chosen by the VQ for prediction of x_k.

Distribution adaptation

Figure 6 shows how the distribution adaptation module works. A time series that obeys to a distinct distribution is given. Minimum and maximum values are not known a priori. The algorithm inside the black box shall form an approximation of the distribution of the given data. The values of the time series are fed to the input of the adaptation module. The density function at the output is represented by a piecewise constant function, see Figure 7.

Usually, a distribution is measured by dividing the whole interesting region into small regions of equal width. The local density inside these regions is calculated from the frequency. The approach presented here has two advantages compared to normal distribution measurement: firstly, there is no a priori knowledge needed concerning the minimum and maximum values of the time series, the algorithm adapts automatically to them. Secondly, the regions are not of equal width and are adapted in order to obtain an optimal split. To achieve a high approximation quality the density function is approximated finer where it is high and coarser where it is low. This is achieved by equal probable regions. The region probability is approxi-

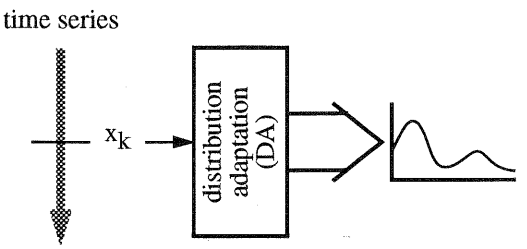


Figure 6 Principle of distribution adaptation.

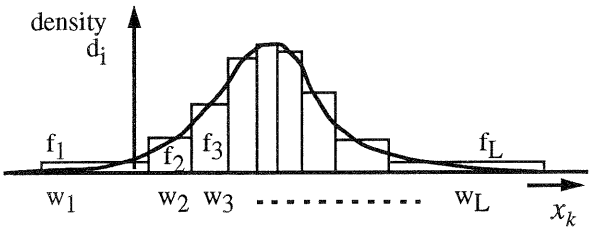


Figure 7 Example of approximated distribution.

mated by the frequency f_i of each region. Figure 7 shows a sample approximation function with L regions. The values w_i denote the width of the regions and d_i the densities, respectively. The correspondence between width, density and region probability is $f_i = d_i \cdot w_i$ for i in $1..L$.

The algorithm is implemented as a non supervised Neural Network with L computing elements. There are no other inputs than the events from the time series for learning. The structure of the NN and the learning rule are described in more detail in Appendix B. In the recall phase, which is needed for distribution prediction, the output of the trained NN is used as the approximation of a density function.

3 PERFORMANCE ANALYSIS

To test the performance of derived models against the original data (i) the distribution and correlation diagrams can be qualitatively compared or (ii) some quantitative tests can be applied.

In this section some statistical performance measures are introduced that are used for quantitative tests.

3.1 Distribution test

To test the distribution the Kolmogorow-Smirnow test is used. This test compares the empirical distribution functions of the underlying time series and the time series generated by the source model. The hypothesis that the source model models the distribution according to a given significance level is accepted or rejected based on a measure that involves the maximal difference between the empirical distribution functions.

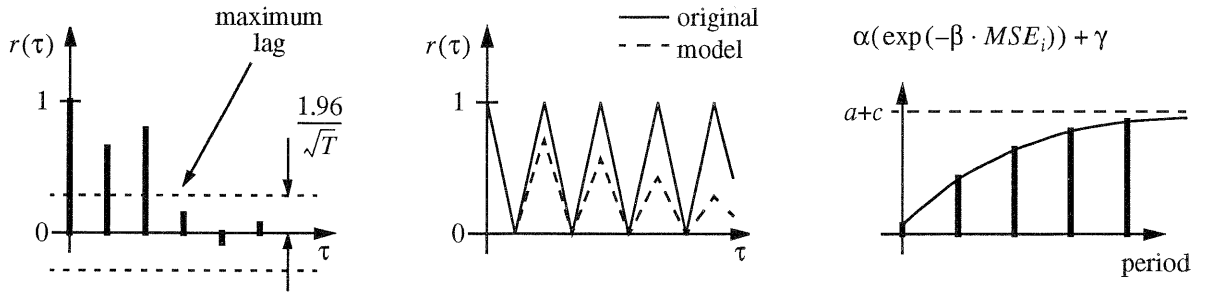
In all examples in this paper a significance level of 0.05 is used which leads to a critical value of $\sqrt{T} \cdot 1.923$ for acceptance of the hypothesis, where T is the sample size of both time series. For the used sample size of 10000 the critical value becomes 192.

3.2 Autocorrelation test

To test the autocorrelation a pragmatic approach is used: the mean square error (MSE) between empirical autocorrelation functions of the underlying time series and the time series generated by the source model. Here once again a distinction has to be made between periodic systems and pure stochastic systems since in case of periodic systems the autocorrelation function does not vanish for higher lags.

Vanishing autocorrelation

The maximal lag for calculating MSE is determined as the lag where the absolute value of the autocorrelation of the underlying time series falls below $1.96/\sqrt{T}$, where T is the sample size,



a.) Determination of maximum lag. b.) Original and model autocorrelation. c.) Exponential approximation.

Figure 8 Tests for correlation.

see Figure 8,a. It is assumed that autocorrelation values less than this value are based on white noise, see Harvey (1993).

In the examples in this paper a threshold of 0.01 for MSE was used for acceptance of autocorrelation.

Periodic time series

In case of periodicity of the underlying time series the autocorrelation does not vanish. On the other hand the autocorrelation function of the generated series decreases approximately exponentially, see Figure 8,b.

That is a feature of the source model. To handle this the MSE is calculated for a number of periods and an exponential function fitted to the resulting error series according to the function

$$\alpha(\exp(-\beta \cdot MSE_i)) + \gamma \ , \tag{12}$$

see Figure 8,c, where MSE_i is the MSE of period i and α, β, γ are parameters to be fitted. The only interesting parameter is β which is used as criterion.

In the examples in this paper a threshold of 0.1 for β was used for acceptance of autocorrelation.

4 SOURCE MODELLING - EXAMPLES

In this section some examples of modelling traffic sources are presented. The comparison between the original time series (reference) and the modelled time series is done by calculating the correlogram for both of them as well as the statistical tests described above.

Markov modulated poisson process - MMPP

The first example is a MMPP process (Markov Modulated Poisson Process) with two states. The MMPP is a frequently used traffic model in telecommunications and represents a source with two activity states. Figure 9 shows a state-transition diagram. The process switches

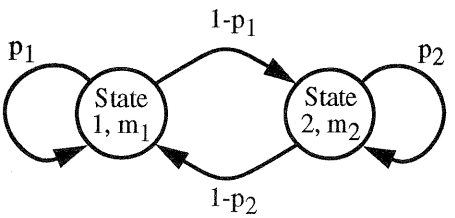


Figure 9 MMPP with 2 states.

between the two states with probabilities $1 - p_1$ and $1 - p_2$, respectively. The expectation of event values in the states are m_1 and m_2. The parameters for the example are $p_1 = p_2 = 0.8$, $m_1 = 0.1$ and $m_2 = 15$.

For distribution prediction N=5 VQ inputs and M=2 VQ units were used, thus having 2 distinct distributions approximated. L=100 segments were used for the approximation of each distribution. Figure 10 shows the resulting distribution functions of the approximation. Note that these distribution functions are not the same than those of the two states of the underlying system since they include the probabilities of state changes, too.

The correlograms in Figure 11 differ to a slight extent, because only two distribution approximation units have been used in this example. See Table 1 for results.

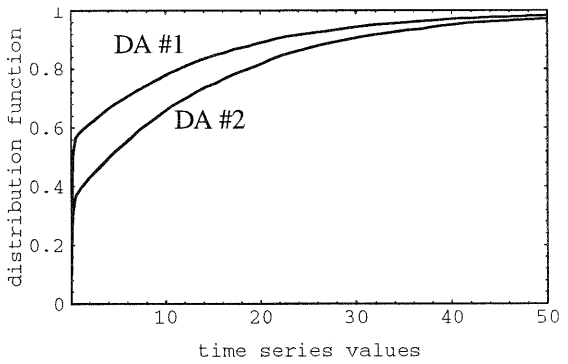


Figure 10 MMPP: Distribution functions.

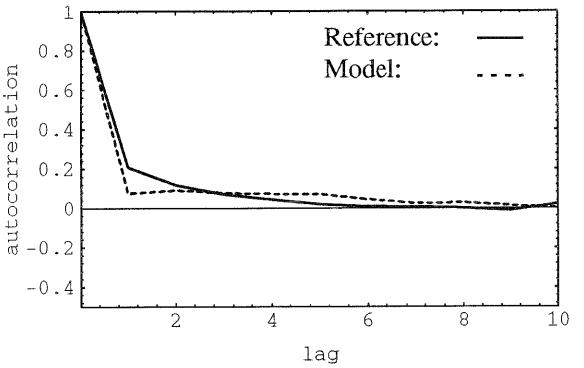


Figure 11 MMPP: Correlogram.

Second order moving average process - MA(2)

The reference data for this test was produced from $y_k = \varepsilon_k + \Theta \cdot \varepsilon_{k-2}$ with $\Theta = 0.2$ and ε being white noise with mean 0 and variance 1.

See Table 1 for model parameter and results. In Figure 12 the good correspondence of the autocorrelations can be seen.

MPEG coded video frames

This example is a real world case. A MPEG coded video sequence from the movie "Star Wars" is used. The sequence consists of the amount of data per video frame after compression.

Figure 13 shows the sequence generated by the MPEG scheme in principle. The large frames are so-called I-frames and comprise a whole picture. The medium-sized frames are P-frames,

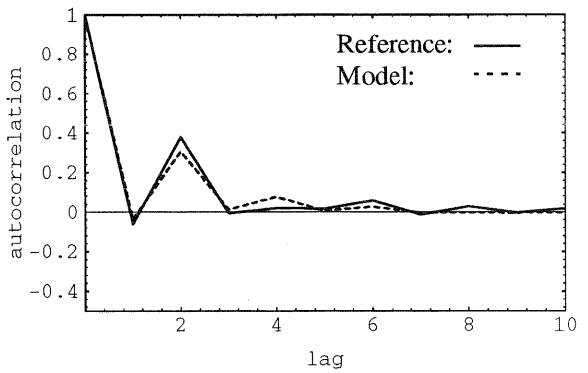


Figure 12 MA(2): Correlogram.

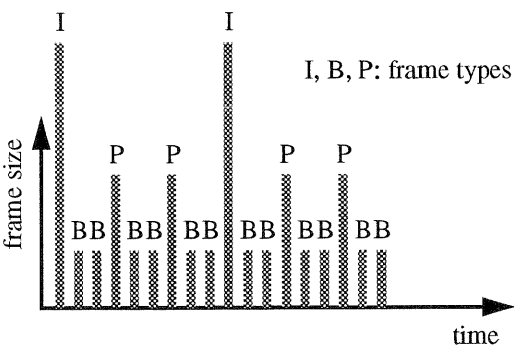


Figure 13 Sequence of MPEG frames.

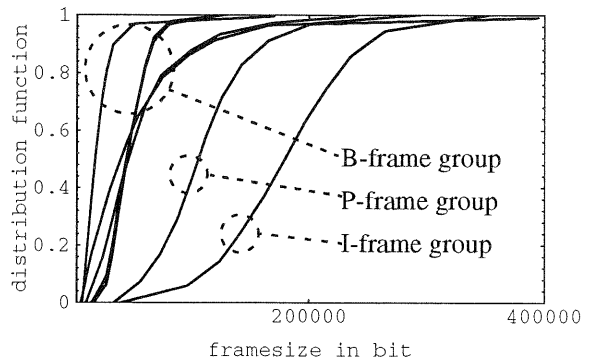


Figure 14 MPEG: Distribution functions.

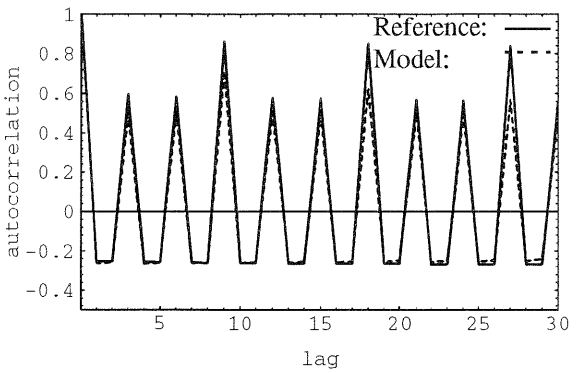


Figure 15 MPEG: Correlation between frames.

the small frames B-frames. The sequence I-B-B-P-B-B-P-B-B- is repeated cyclically and is defined by the MPEG parameters (more detailed information can be found in Le Gall (1991)). The size of the I-, B-, and P-frames is distributed according to the underlying video scenes. Due to this cyclical behavior the correlogram of MPEG-coded video data has a characteristic form as shown in Figure 15.

Figure 14 shows the resulting distribution approximations that are learned by the VQ and the DA's during the adaptation process. It can be seen that there are different groups of distributions. Each group represents a different frame type (I, B or P) and the occurrence of the different distribution types obeys the same rule than the I-B-P-sequence above.

In Figure 15 it can be seen that the model fits the reference quite well. Model parameters and further results are contained in Table 1.

The long term correlation is poor in this example but it could be further improved by increasing the number N of model inputs.

Table 1 Examples

	N	M	L	*distribution test (value should be < 192)*	*correlation test*
MMPP	5	2	100	185	0.002 (MSE)
MA(2)	3	25	10	189	0.0026 (MSE)
video frames	9	15	10	165	0.04 (periodic)

5 FURTHER DEVELOPMENT

This work is still in a preliminary state and extensions are under development. Future extensions will include:

- Control inputs for source models. This allows hierarchical models for different time scales. In case of video data a higher model may be responsible to model video scenes whereas a lower model is responsible to model the frame sizes.
- Multivariate time series.

This should be easily done by extending the input vector with control lines and values from other time series. During the adaptation procedure the distribution approximation can be adapted independently for every input stream.

6 SUMMARY

In this paper a new algorithm is presented that identifies arbitrary time series that may contain seasonalities. It consists of a vector quantizer that reduces the complexity of the input data and a special Neural Network type that is able to learn distributions.

The model parameters for both the vector quantizer and the Neural Network are automatically derived by the learning process if an adequate topology is chosen.

A performance analysis is presented and some examples demonstrate the usability of the method.

7 REFERENCES

Bagchi, A. (1993) *Optimal Control of Stochastic Systems*. Prentice Hall.

Brillinger, D. (1992) *New Directions in Time Series Analysis, Part I+II*. Springer.

Chakraborty, K., Mehrotra, K., Mohan, C. K. and Ranka, S. (1992) Forecasting the Behavior of Multivariate Time Series Using Neural Networks. *Neural Networks*, vol. 5, 961-70.

Deppisch, J., Bauer, H. U. and Geisel, T. (1994) Hierarchical training of neural networks and prediction of chaotic time series, in *Artificial Neural Networks: Forecasting Time Series* (ed. V. R. Vemuri, R. D. Rogers), IEEE Computer Society Press, 66-71.

Gruenenfelder, R., Cosmas J. P. and Odinma-Okafor, A. (1991) Characterization of Video Codecs as Autoregressive Moving Average Processes and Related Queueing System Performance. *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 3, 284-93.

Hamilton, J. D. (1994) *Time Series Analysis*. Princeton University Press.

Harvey, A. C. (1993) *Time Series Models*. Harvester Wheatsheaf.

Hecht-Nielsen, R. (1989) *Neurocomputing*. Addison-Wesley.

Hudson, J. L., Kube, M., Adomaitis, R. A., Kevrekidis, I. G., Lapedes, A. S. and Farber, R. M. (1994) Nonlinear Signal Processing and System Identification: Applications to Time Series from Electrochemical Reactions, in *Artificial Neural Networks: Forecasting Time Series* (ed. V. R. Vemuri, R. D. Rogers), IEEE Computer Society Press, 36-42.

Janacek, G. (1993) *Time Series*. Ellis Horwood.

Law, A. M. and Kelton, W. D. (1991) *Simulation Modelling & Analysis*. McGraw-Hill.

Le Gall, D. (1991) MPEG: A Video Compression Standard for Multimedia Applications, *Communications of the ACM*, vol. 34, no. 4, 46-58.

Lowe, D. and Webb, A. R. (1994) Time series prediction by adaptive networks: a dynamical systems perspective, in *Artificial Neural Networks: Forecasting Time Series* (ed. V. R. Vemuri, R. D. Rogers), IEEE Computer Society Press, 12-9.

Mozer, M. C. (1993) Neural Net Architectures for Temporal Sequence Processing, in *Time Series Prediction: Forecasting the Future and Understanding the Past* (ed. A. S. Weigend), Addison-Wesley, 243-64.

Papoulis, A. (1984) *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill.

Scargle, J. D. (1992) Predictive Deconvolution of Chaotic and Random Processes, in *New Directions in Time Series Analysis, Part I* (ed. D. Brillinger), Springer, 335-56.

Tang, Z., de Almeida, C. and Fishwick, P. A. (1994) Time series forecasting using neural networks vs. Box-Jenkins methodology, in *Artificial Neural Networks: Forecasting Time Series* (ed. V. R. Vemuri, R. D. Rogers), IEEE Computer Society Press, 20-7.

Tarraf, A. A., Habib, I. W. and Saadawi, T. N. (1993) Neural Networks for ATM Multimedia Traffic Prediction. *Proceedings of IWANNT '93*, 85-91.

Tarraf, A. A., Habib, I. W. and Saadawi, T. N. (1994) A Novel Neural Network Traffic Enforcement Mechanism for ATM Networks. *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 6, 1088-96.

Wan, E. A. (1993) Time Series Prediction Using a Connectionist Network with internal Delay Lines, in: *Time Series Prediction: Forecasting the Future and Understanding the Past* (ed. A.

S. Weigend), Addison-Wesley, 195-217.
Weigend, A. S. (1993) *Time Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley.

APPENDIX A:VECTOR QUANTIZER

The VQ algorithm described here is closely related to the Kohonen learning rule as proposed in Hecht-Nielsen (1989).

The vector quantizer consists of M units that receive the following input (Euclidean distance between $\boldsymbol{i} = (i_1, i_2, \ldots, i_N)$ and internal weight vectors $\boldsymbol{w}_i = (w_{i1}, w_{i2}, \ldots, w_{iN})$, i=1..M) that belong to the units, see Figure 16:

$$d_i = D(\boldsymbol{w}_i, \boldsymbol{i}) = \sqrt{(w_{i1} - i_1)^2 + \ldots + (w_{iN} - i_N)^2} \tag{13}$$

A competition takes place between the units. The unit with the lowest value $d_i - b_i$ is the winner and its output z_i is set to 1. The outputs of all other units are set to 0. The value b_i is a bias term that ensures that the frequency of winning the competition becomes $1/M$ for all units after some learning.

The rule for calculating the bias is

$$b_i = \gamma \cdot \left(\frac{1}{M} - f_i\right), \quad i\text{=}1..M \ (\gamma \text{ typically } 10) \tag{14}$$

After the competition the weight vector of the winner unit and the f_i of all units are modified:

$$\boldsymbol{w}_i^{new} = \boldsymbol{w}_i^{old} + \alpha(\boldsymbol{i} - \boldsymbol{w}_i^{old}) \tag{15}$$

$$f_i^{new} = f_i^{old} + \beta(z_i - f_i^{old}), \quad i\text{=}1..M \ (\beta \text{ typically } 0.0001) \tag{16}$$

The learning rate α decreases exponentially during the learning process (typically from 0.05 to 0.001).

APPENDIX B:NEURAL NETWORK ALGORITHM

In this Appendix the learning rule of the new Neural Network algorithm is shortly described. For the meaning of some variables refer to Figure 7 in Section 2 and to Table 2.

The first step is to determine the segment i the event e_k falls into. Next segment frequency and width are adapted.

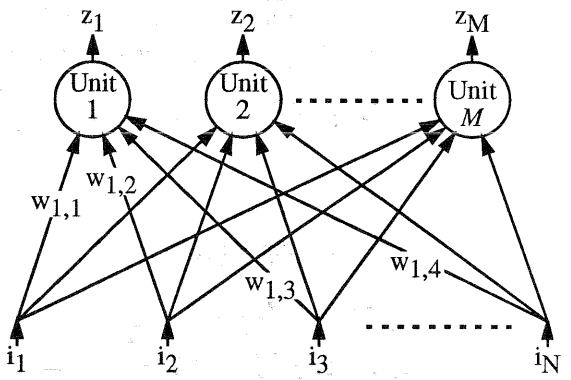


Figure 16 Vector quantizer.

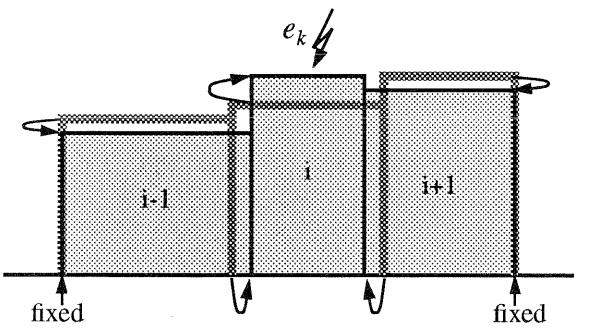


Figure 17 Event in segment i.

Principle

Events have only an effect in one segment and its direct left and right neighbors (segments i, i-1, i+1). Therefore the algorithm has a high locality and can be easily implemented in parallel.

Table 2 Variables for Neural Network algorithm

variable	*description*
e_k	Event at time k
$w_{i,k}$:	Width of segment i
$d_{i,k}$:	Density of segment i
$f_{i,k}$:	Frequency of segment i
L	Number of segments for approximation

Adaptation of segment width and position

The adaptation of segment width consists of the distribution of an "amount of frequency" of segment i to its neighboring segments. The density of segment i is kept constant as well as the borders of the neighbors. Adaptation takes place by means of the change of the width of segments i, i-1, i+1 and the change of density of segments i-1 and i+1. The width of segment i is reduced to enhance the resolution in areas of higher event frequency. This procedure leads to nearly the same probability $1/L$ of each segment.

The marginal segments (segments 1 and L) have to be treated in a special way. Their outer borders have to adapt to the minimum and maximum values of the underlying distribution.

The "amount of frequency" mentioned above is governed by a learning parameter λ that decreases exponentially during learning (typically from 0.1 to 0.001).

Adaptation of segment frequency

The adaptation of segment frequency firstly corrects some errors possibly made in the adaptation of segment width and secondly provides a fine tuning of the density function. The adaptation of segment frequency is performed for more cycles than the adaptation of segment width. So even if the first step leads to non-optimal density function this is corrected in the second step.

The frequency of events in the distinct segments is calculated as exponentially weighted mean value of all events falling in one segment:

$$f_{i,k} = \alpha \cdot \sum_{j=0}^{k} (1-\alpha)^j \cdot t_{i,k-j} \tag{17}$$

with

$$t_{i,k} = \begin{cases} 1 & \text{if event } e_k \text{ in area } i \\ 0 & \text{if event } e_k \text{ not in area } i. \end{cases} \tag{18}$$

Since the $t_{i,k}$ are zero for k negative equation (17) can be rewritten as a recursion:

$$f_{i,k} = \alpha \cdot t_{i,k} + (1-\alpha) \cdot f_{i,k-1} \tag{19}$$

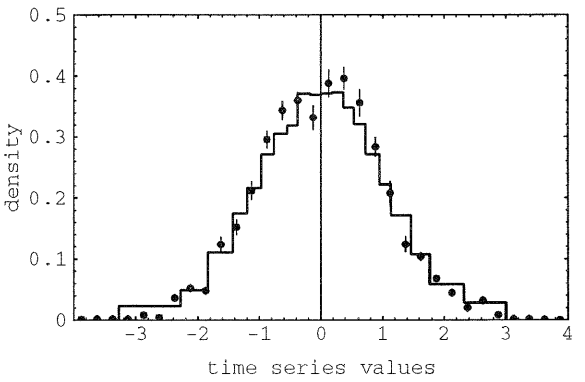


Figure 18 Density approximation of normal distributed time series.

The factor α determines the contribution of the actual event $t_{i,k}$ to the sum. Like for the adaptation of the segment width a slightly modified method is necessary to adapt the segment frequency of boundary segments. The value of α depends on the number of samples available for learning. The combined effects of both adaptations can be seen in Figure 17.

Example

An example for the adaptation capabilities of the Neural Network is shown in the sequel. A normally distributed time series with mean zero and variance 1 is used as training data. Figure 18 shows the original density function (dots) with 95% confidence intervals (irregularity in shape stems from non-ideal random number generator and not much adaptation cycles) and the approximation with L=20 segments (solid line). Note the decreasing width with increasing density.

Error approximation

The resulting distribution approximation has an approximation error. The error is computed as the integral over the squared difference of original distribution f and density approximation d_i. After some simplification and assuming that all regions have the same probability $1/L$ this leads to

$$E = \int_{-\infty}^{\infty} f^2(x)dx - \frac{1}{L} \cdot \sum_{i=1}^{L} d_i. \tag{20}$$

8 BIOGRAPHY

Markus D. Eberspächer was born in 1963 and studied Electrical Engineering at the University of Stuttgart where he received the Dipl.-Ing. degree in 1991. Since 1991, he is a member of the scientific staff at the Institute of Communications Switching and Data Technics, University of Stuttgart (Prof. Kühn). His interests include Neural Networks and Fuzzy Control in telecommunication networks.