10 Data Mining Techniques for Designing Efficient Neural Network Time Series Predictors

RADU DROSSU
rdrossu@hotmail.com

ZORAN OBRADOVIĆ
zoran@eecs.wsu.edu

School of Electrical Engineering and Computer Science
Washington State University, Pullman, Washington, 99164-2752

Abstract

A real-life time series prediction system is usually subject to two constraints - accuracy and time, meaning that a sufficiently accurate prediction has to be provided in an imposed time frame. The objective of this article is to demonstrate that the knowledge obtained through relatively simple data mining can be embedded into a neural network time series predictor in order to both reduce its design time and improve its accuracy. Direct knowledge embedding methods, based on information theoretical, dynamical system analysis, and stochastic modeling are discussed. It is illustrated that direct methods can produce a wealth of prior information regarding the choice of an appropriate neural network architecture, data sampling rates, as well as starting values for the model parameters, which otherwise have to be found as a result of costly trial-and-error procedures. In addition to the direct knowledge embedding, the article also discusses indirect embedding methods which exploit known properties of the target function and non-stationarity detection techniques. The use of known properties of the target function can enlarge scarce data sets or enforce a more accurate learning through constrained optimization. Non-stationarity analysis can considerably improve the computational efficiency of time series forecasting by avoiding the neural network model re-design more often than needed.

10.1 Introduction

The outcomes of a phenomenon over time form a *time series*. Time series are encountered in sciences as well as in real life. The voltage measured every second across a resistor in an electrical circuit, the number of cars passing a marker on a highway every minute, the yearly power consumption of the United States, the

hourly exchange rate of German mark versus U.S. dollar, the daily car production of Chrysler corporation are just a few examples of time series. Although sometimes outcomes of processes described through mathematical closed forms (known deterministic functions) are also viewed as time series, most commonly time series are the result of unknown or not completely understood processes. Therefore, more formally, a time series $\{x_t\}$ can be defined as a function x of an independent variable t, stemming from an unknown process. Its main characteristic is that its evolution can not be described exactly as in the case of a known deterministic function of t.

It is human nature to have the desire to know in advance what is likely to happen in the future. The observation of past outcomes of a phenomenon in order to anticipate its future behavior represents the essence of *forecasting (prediction)*. If a mathematical model describing a studied phenomenon is known, forecasting becomes a trivial and degenerate task. However, if a model of the phenomenon is either unknown or incomplete, different attempts can be made for predicting its future evolution. A typical approach is to try to predict by constructing a model which takes into account solely previous outcomes of the phenomenon while ignoring any other additional exterior influence. Alternatively, a prediction model can be constructed which incorporates all the factors which presumably influence the process under consideration. For example, the simplest attempt to predict the U.S. power consumption would be by using a prediction model based just on previous values of power consumption which neglects any other information that might also be available. On the other hand, we could construct a model which incorporates additional variables that presumably influence the power consumption like temperature, time of day, season, etc. The choice of one or the other of the two approaches is problem dependent and care must be taken when developing a prediction model in order not to include variables that do not bear any influence on the phenomenon under study, since these would merely act as input noise.

Real-life time series are often the result of complex and insufficiently understood interdependencies. Hence, prediction models make use of incomplete information, while other factors not included in the models act as noise. In addition, real-life time series are often non-stationary, meaning that the data distribution is changing over time. Therefore, for non-stationary domains, a single model built on a certain data segment and used for all subsequent predictions is generally inadequate. A straightforward attempt is to stationarize the data by performing a de-trending preprocessing (e.g., a first or a second order discrete differentiation). More sophisticated methods provide solutions for certain types of non-stationarity (e.g., a reversible power transformation is successfully used to stabilize the variance of a series affected by a strong trend that cannot be removed by differentiation (Abecasis and Lapenta, 1996)). However, not all non-stationary processes can be stationarized through data preprocessing. Forecasting such processes requires *on-line learning techniques*, where a given model is used for a limited time and a new model is constructed whenever a change of the underlying data distribution is detected.

The two issues which have to be addressed by any time series prediction system are *accuracy* and *time*, meaning that a sufficiently accurate prediction has to

be provided in an imposed time frame. Quite often these two constraints are contradictory, signifying that usually, given more time for designing a prediction model a better accuracy could be achieved. If time is not an issue, like when predicting the yearly power consumption of the United States, accuracy would be the only constraint that the design process has to deal with. In these cases multilayer perceptron neural networks are often used in practice. Their popularity is due to their universal approximation capabilities, meaning that they can represent non-linear complex functions to any desired accuracy (Cybenko, 1989), and to their significantly better scaling with the dimensionality of the input space as compared to traditional approximation techniques (Barron, 1993).

However, the design of an appropriate neural network for time series prediction problems with high data arrival rates (e.g., Internet traffic predictions or financial intra-day predictions) can be a challenging task, due to time consuming trial-and-error architecture selection, non-linear parameter optimization and the need to devise new prediction models whenever the underlying data distribution changes. For these reasons, any prior knowledge that could be extracted from a time series under study can dramatically decrease the design time of a predictor and also improve its prediction accuracy significantly.

This article discusses two categories of prior knowledge extraction techniques, which are embeddable into neural network prediction models. The first category, discussed in Section II and denoted as *direct knowledge embedding* encompasses information theory, non-linear dynamics, and stochastic analysis. These techniques of exploratory data analysis can provide prior knowledge regarding appropriate neural network architecture, initial network parameters and adequate data sampling rate. A real-life time series (compressed video traffic data), as well as an artificial, non-linear, chaotic time series (Mackey-Glass data) are used to illustrate the embedding of prior knowledge extracted from stochastic analysis into the neural network design process. The second category, *indirect knowledge embedding*, addressed in Section III, includes the use of known properties of target functions and non-stationarity detection. The use of known properties of the target function can enlarge scarce data sets by creating additional artificial training examples, or enforce a more accurate learning through constrained optimization. Non-stationarity analysis can considerably improve the computational efficiency of time series forecasting by avoiding the neural network model re-design more often than needed. Therefore, different distribution-change signaling techniques for deciding whether to reuse "trusted" models or retrain new ones are discussed and compared on low-noise and high-noise, artificially generated, non-stationary time series.

10.2 Direct Information Extraction Procedures

10.2.1 Information Theory.

It has been hypothesized (Barlow, 1989) that the redundancy between different input signals allows the brain to discover their statistical relationships, and to use them for object recognition and associative learning. A representation has further been proposed in which the individual signals are statistically independent, as being the most appropriate for storing statistical information. In order to acquire the statistical independence (*factorial code*) of the input signals, a multiple-stage learning process can be employed in which every stage reduces the redundancy between its input signals generating more decorrelated output signals. This redundancy reduction strategy is applied in an unsupervised fashion to the problem of eliminating redundancies from English text (Redlich, 1993a).

In order to make redundancy reduction effective for pattern recognition or time series prediction problems, supervision has to be incorporated into the learning process. The goal of the redundancy reduction process, denoted as *factorial learning* (Redlich, 1993b), is to approximate the joint probability $P(x_1, x_2, \ldots, x_d) = P(\bar{x})$, where d represents the number of input signals, as a product of the individual probabilities, $P(x_1), P(x_2), \ldots, P(x_d)$. The individual learning stages represent better and better approximations to the joint probability. In order to evaluate the quality of the factorial approximation at different learning stages, we can make use of the entropy function. Considering that each learning stage represents an $\mathcal{R}^d$ to $\mathcal{R}^d$ mapping, we can use as a cost function the sum of output entropies (Redlich, 1993b), expressed as

$$E = \sum_{i=1}^{d} H_i, \tag{10.1}$$

where the individual entropies are computed as

$$H_i = -\sum_{x_i} P(x_i) \log_2 \left[P(x_i)\right], \tag{10.2}$$

with the sum running over all the discretized values of the output variable x_i. This cost function is minimal when the code is factorial (Redlich, 1993b), so that reducing E in stages improves the factorial representation of the joint probability. In order to be able to obtain an approximation to the global joint probability after a number of stages, which is solely a function of probabilities at the output level of the last stage, we need to impose that information is preserved from one learning stage to the next. Denoting by $\bar{x}$ the input vector to a certain learning stage and by $\bar{y}$ the corresponding output vector, the information preservation condition can be written as

$$H\left[P(\bar{x})\right] = H\left[P(\bar{y})\right], \tag{10.3}$$

with $H[\cdot]$ denoting the total entropy at input or output level. With this additional constraint, minimizing E will produce a factorial approximation to the input joint probability, due to the independence bound on the entropy, according to which

$$H \leq \sum_i H_i, \tag{10.4}$$

with equality only when the code is factorial. A network named *almost reversible cellular automata* (ARCA) has been proposed by Redlich (Redlich, 1993b), which can be used, both in an unsupervised and in a supervised fashion, for factorial learning. The ARCA network proposed for supervised learning, a viable alternative to multi-layer perceptron networks, constructs additional computational layers as needed, in a fashion similar to cascade correlation networks (Fahlman and Lebiere, 1990). However, its applicability appears to be restricted to classification problems. Therefore, we will illustrate an approach proposed by Deco and collaborators (Deco and Schurmann, 1995; Deco and Brauer, 1995), similar to the unsupervised ARCA networks, which could be incorporated as a preprocessing module in a time series prediction system.

The goal is to construct a one stage neural network which attempts to statistically decorrelate the components of its output vector (Deco and Schurmann, 1995; Deco and Brauer, 1995). In order to impose the constraint of no information loss from input to output, we can use the mutual information between input and output as a measure of information transmission, defined as (Deco and Obradovic, 1996)

$$I(\bar{y};\bar{x}) = H(\bar{x}) - H(\bar{x}|\bar{y}), \tag{10.5}$$

with $H(\bar{x})$ denoting the input entropy and $H(\bar{x}|\bar{y})$ denoting the conditional entropy of $\bar{x}$ given $\bar{y}$. The output entropy satisfies the inequality (Deco and Schurmann, 1995)

$$H(\bar{y}) \leq H(\bar{x}) + \int P(\bar{x}) \ln \left[\det \left[\frac{\partial \bar{T}}{\partial \bar{x}} \right] \right] d\bar{x}, \tag{10.6}$$

with equality holding only when the transformation $\bar{T}$ is bijective, thus reversible. Bijectivity is therefore satisfied if the Jacobian of the transformation has a unit determinant,

$$\det \left[\frac{\partial \bar{T}}{\partial \bar{x}} \right] = 1. \tag{10.7}$$

The architecture proposed in (Deco and Schurmann, 1995; Deco and Brauer, 1995), which fulfills the previous requirement and has the same structure as the ARCA network is presented in Fig. 10.1.

The transformation computed by this network can be expressed analytically as

$$y_i = x_i + f_i(x_1, \ldots, x_j, \bar{w}_i), \quad j < i, \tag{10.8}$$

with $\bar{w}_i$ denoting some parameter vector which intervenes in the computation

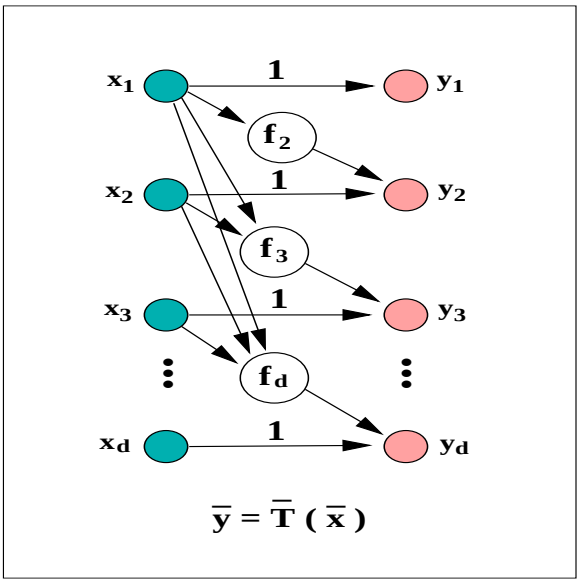

Figure 10.1 Information Preserving Transformation

of function f_i. It is obvious that the Jacobian of the network transformation is an upper triangular matrix, with all the diagonal elements equal to 1, which yields a determinant equal to unity and thus represents a transformation without information loss. The specific form assumed for the functions f_i in (Deco and Schurmann, 1995; Deco and Brauer, 1995) is polynomial, resulting in network outputs computed according to

$$y_i = x_i + \sum_{j=1}^{i-1} w_{ij} x_j + \sum_{j,k=1}^{i-1} w_{ijk} x_k x_j + \ldots \tag{10.9}$$

Viewed as a preprocessing module for a time series prediction problem, $x_1, \ldots, x_d$ would represent d previous samples from a univariate time series and $y_1, \ldots, y_d$ the outputs that tend to be decorrelated as a result of the learning process.

In order to obtain the decorrelation of the output signals, Deco and the following alternatives is proposed (Deco and Schurmann, 1995):

1. the minimization of an upper bound on the mutual information between the components of the output vector;

2. cumulant expansion of the output distribution followed by imposing the independence condition.

Thus, the decorrelation can be achieved by gradient descent error minimization using either the cost function (see Appendix A),

$$E = \sum_{i=1}^{d} \ln(\sigma_i^2), \tag{10.10}$$

with σ_i^2 being the variance of component i of the output vector, or (see Appendix B),

$$\begin{aligned} E &= \alpha \sum_{i<j} < y_i y_j >^2 + \beta \sum_{i<j\leq k} < y_i y_j y_k >^2 \\ &+ \gamma \sum_{i<j\leq k\leq l} < y_i y_j y_k y_l >^2 \\ &+ \delta \sum_{i<j} \left(< y_i^2 y_j^2 > - 3 < y_i^2 >< y_j^2 > \right)^2 , \end{aligned} \tag{10.11}$$

with

$$< x_1 x_2 \ldots x_n >= \int x_1 x_2 \ldots x_n p(\bar{x})\, d\bar{x}, \tag{10.12}$$

and α, β, γ, and δ, representing the inverses of the number of elements in their corresponding summations.

The decorrelation transformation implemented using the previously introduced neural network and either of the two cost functions presented leads also to an implicit determination of an appropriate embedding dimension (number of neural network inputs), by discarding the outputs with a variance which is below an imposed threshold. Set in the unsupervised learning framework, the information preserving transformation is not directly applicable to prediction problems. Nevertheless, it could be used as a building block inside a prediction system, whose general structure could be as follows:

- decorrelation module (performs previously described decorrelation transformation);
- predictor module (feedforward or recurrent neural network performing a "spatial" prediction in the transformed space obtained after decorrelation);
- inverse transformation module (computes the inverse of the transformation computed by the decorrelation module, in order to provide an informative prediction in the original data space).

10.2.2 Dynamical System Analysis.

Dynamical system analysis provides a potential means of obtaining information regarding an appropriate neural network architecture (more precisely, regarding the number of neural network inputs), as well as regarding an appropriate data sampling rate in prediction problems with prediction horizon larger than one (Lapedes and Farber, 1987).

A dynamical system is described in terms of a set of *state variables*, whose values at any given time t are assumed to contain sufficient information to describe the future evolution of the system (Haykin, 1994). The *state vector* for a univariate time series can be constructed from delayed samples of the series $\bar{x}(t) = [x(t), x(t-\tau), \ldots, x(t-(M-1)\tau)$. Assuming that the system can be represented in terms of M state variables which can be grouped together to form a state vector, $\bar{x}(t) = [x_1(t), \ldots, x_M(t)]$, the evolution of a dynamical system can be

observed as a *trajectory* in the M-dimensional state space, also known as the *phase space* of the system. The phase space can be a Euclidean space or a subset thereof. It can also be a non-Euclidean space such as a sphere or a torus, or some other differentiable manifold (Haykin, 1994). Many dynamical systems can be described by a set of differential equations

$$\frac{d}{dt}\bar{x}(t) = V\left(\bar{x}(t)\right), \tag{10.13}$$

where $V(\cdot)$ is a non-linear (vector) function of the state vector. The family of trajectories which result when starting from different initial conditions, form the *phase portrait* of the system. The phase portrait includes all the points in the phase space where $V(\bar{x})$ is defined.

In general, a *dissipative system* (a system which "looses" energy), is characterized by the convergence of its trajectories in phase space onto manifolds of lower dimensionality, $m < M$, which form the dynamical system's *attractor* (Haykin, 1994). Attractors represent equilibrium states of a dynamical system which can be observed on experimental data. Different methods are employed in practice for determining the dimensionality of a system's attractor, the most representative ones, probably, being based on estimating either the (generalized) *Renyi dimensions*, or the *Lyapunov exponents*.

The Renyi dimensions represent a spectrum of dimensions, $D_0 \geq D_1 \geq D_2 \geq \ldots$, defined as (Pineda and Sommerer, 1993),

$$D_q = \frac{1}{q-1} \limsup_{\epsilon \to 0} \frac{\log(\sum_i p_i^q)}{\log(\epsilon)}. \tag{10.14}$$

Their computation assumes that the attractor is covered by M-dimensional boxes of side length ϵ, and p_i represents a measure of the attractor in box i, with the sum taken only over occupied boxes (boxes containing at least one data point). For finite data sets, the p_i's can be approximated as n_i/n, with n_i being the number of data points in the i-th box and n representing the total number of data points. The first three generalized dimensions, also known as *capacity dimension*, *information dimension* and *correlation dimension*, respectively, can further be expressed as

$$D_0 = \limsup_{\epsilon \to 0} \frac{\log N(\epsilon)}{\log(1/\epsilon)}, \tag{10.15}$$

with $N(\epsilon)$ representing the number of non-empty boxes,

$$D_1 = \limsup_{\epsilon \to 0} \frac{-\sum_i p_i \log p_i}{\log(1/\epsilon)}, \tag{10.16}$$

and

$$D_2 = \limsup_{\epsilon \to 0} \frac{-\log\left(\sum_i p_i^2\right)}{\log(1/\epsilon)}. \tag{10.17}$$

The technique proposed at (Pineda and Sommerer, 1993) for computing D_0, D_1 and

D_2 starts by plotting the numerators in the formulas for D_0, D_1, and D_2 versus $1/\epsilon$ for different space dimensions M, yielding three families of curves for D_0, D_1, and D_2, respectively, represented on separate plots. For each curve (corresponding to an individual value for M), the slope of its linear region before saturation is determined. These slopes are then plotted versus space dimension and the value at which the resulting curve saturates, provides the value for the generalized dimension under consideration. To check whether the data support the dimension computations, a final graph should be provided which contains the slope versus space dimension curves for all of D_0, D_1, and D_2. The fulfillment of the condition $D_0 \geq D_1 \geq D_2$ at any space dimension supports the hypothesis that enough data has been provided in order to compute the generalized dimensions. A lack of saturation of the previously mentioned curves can be an indication of either a stochastic (non-deterministic) process, or of insufficient data. Different alternate measures exist, which allow the distinction between deterministic chaos and random noise (Grassberger and Procaccia, 1983). Any of D_0, D_1 or D_2 could in principle be used to estimate m, since they are usually very close, but in practice D_1 is the most commonly used. Based on Takens' theorem (Takens, 1981), an estimate of the dimension m of the time series' attractor can be used to construct a multi-layer perceptron neural network of $2m + 1$ external units (Lapedes and Farber, 1987).

Instead of computing the generalized dimensions, we can alternatively compute the Lyapunov exponents using the available experimental data (Wolf et al., 1985). Loosely speaking, the Lyapunov exponents represent measures of change for geometric bodies of increasing dimensionality, as produced by the trajectories of the dynamical system. Thus, the first Lyapunov exponent, λ_1 measures the average logarithmic growth of the relative error per iteration between two initial conditions on neighboring trajectories of a dynamical system (Jurgens and Saupe, 1992). In other words, e^{λ_1} represents the maximal average factor by which an error between neighboring trajectories is amplified. Expressed mathematically, the first Lyapunov exponent is given by (Jurgens and Saupe, 1992)

$$\lambda_1 = \lim_{n \to \infty} \lim_{E_0 \to 0} \frac{1}{n} \sum_{k=1}^{n} \log \left| \frac{E_k}{E_{k-1}} \right|, \tag{10.18}$$

with E_0 representing the initial error and E_k/E_{k-1} denoting the error amplification from one step to the next. The second Lyapunov exponent, λ_2, represents a measure of how an area is changed along the "flow" of the dynamical system. Expressed differently, $e^{\lambda_1+\lambda_2}$ represents the maximal average factor by which an area changes. Similarly $e^{\lambda_1+\lambda_2+\lambda_3}$, where λ_3 represents the third Lyapunov exponent, expresses the maximal average factor by which a volume changes. The process of determining Lyapunov exponents can continue with exponents of higher order, all of them being subject to the ordering

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots, \tag{10.19}$$

with positive exponents standing for expansion along a certain direction and

negative exponents denoting contraction along a direction.

Finally, the *Lyapunov dimension* can be computed as

$$D_L = \begin{cases} i + \frac{1}{|\lambda_{i+1}|} \sum_{k=1}^{i} \lambda_k, & \text{if } \lambda_1 \geq 0 \\ 0, & \text{otherwise,} \end{cases} \tag{10.20}$$

with i being the maximum integer with $\lambda_1 + \ldots \lambda_i \geq 0$. In theory, the Lyapunov exponents and the Lyapunov dimension are computed in a relatively straightforward manner for a continuous time series (generated from a set of differential equations). However, in practice, when dealing with a discrete time series, the effect of lacunarity (finite amount of data) has a negative impact on the accuracy of the results for both generalized dimensions and Lyapunov exponents. The Lyapunov dimension can be related to the information dimension in accordance to the *Kaplan-Yorke conjecture*, which claims their equality. Hence, denoting by m the dimension (information dimension or Lyapunov dimension) of a given time series, a potentially adequate multi-layer perceptron for predicting the time series should have a number of input units which is equal to $2m + 1$.

Dynamical system analysis can further provide an indication of an appropriate data sampling rate (time delay), to be used in prediction problems with larger prediction horizon (predicting further into the future) (Liebert and Schuster, 1989; Pineda and Sommerer, 1993). Pineda and Sommerer (Pineda and Sommerer, 1993) consider the original time series, $\{x_t\}$, as well as its time-delayed counterpart, $\{x_{t-\tau}\}$ (with time origin shifted by τ), which are discretized in units of ϵ (e.g., bits). Consequently, denoting by X the random variable associated with the process values for $\{x_t\}$ and by Y the random variable corresponding to the process values for $\{x_{t-\tau}\}$, we can define the discrete probabilities

$$P_X(x) = \text{Prob}\{X = x\}, \tag{10.21}$$

$$P_Y(y) = \text{Prob}\{Y = y\}. \tag{10.22}$$

and

$$P_{XY}(x, y) = \text{Prob}\{X = x \text{ and } Y = y\}. \tag{10.23}$$

Taking into consideration the quantized process values, we can define the scale-dependent entropies (Deco and Obradovic, 1996)

$$H_X(\epsilon) = -\sum_i P_X(x_i) \log_2 P_X(x_i), \tag{10.24}$$

$$H_Y(\epsilon) = -\sum_i P_Y(y_i) \log_2 P_Y(y_i), \tag{10.25}$$

as well as the scale-dependent cross-entropy,

$$H_{XY}(\epsilon) = -\sum_i P_{XY}(x_i, y_i) \log_2 P_{XY}(x_i, y_i), \tag{10.26}$$

with the sums taken over all the occupied one-dimensional and two-dimensional boxes, respectively. Additionally, we can also define the mutual information between the random variables X and Y as

$$M_{XY}(\epsilon) = H_X(\epsilon) + H_Y(\epsilon) - H_{XY}(\epsilon). \tag{10.27}$$

Making explicit the dependence on the delay τ, we can rewrite the mutual information in the approximate form,

$$M_{XY}(\epsilon, \tau) = [2D_X - D_{XY}(\tau)] \log_2 \epsilon, \tag{10.28}$$

with D_X being the information dimension D_2 computed for a space dimension $M = 1$ and D_{XY} being the information dimension computed for $M = 2$. This expression allows the definition of a box size independent mutual information as,

$$D_\mu(\tau) = 2D_X - D_{XY}(\tau). \tag{10.29}$$

Finally, the optimal value of τ is chosen to be the one corresponding to the first minimum of the mutual information between the actual time series and the delayed one.

Although prone to lacunarity effects and not effective for stochastic processes, dynamical system analysis is still useful in providing an indication of whether the underlying time series stems from a deterministic or from a stochastic process. This information is especially useful, since domain-specific analysis techniques are likely to provide more accurate results than general ones.

10.2.3 Stochastic Analysis

Time series prediction is traditionally approached using stochastic methods (Box et al., 1994). A popular and theoretically well founded stochastic model for a stationary time series is the autoregressive moving average model of orders p and q, denoted as ARMA(p,q), which describes the process value as a weighted sum of p previous process values and the current as well as q previous values stemming from a random process. Formally, the stationary ARMA(p,q) process with zero mean $\{x_t\}$ is represented as

$$x_t = \varphi_1 x_{t-1} + \ldots + \varphi_p x_{t-p} + a_t + \psi_1 a_{t-1} + \ldots + \psi_q a_{t-q}, \tag{10.30}$$

where $x_{t-1}, x_{t-2}, \ldots, x_{t-p}$ represent the process values at p previous time steps, $a_t, a_{t-1}, \ldots, a_{t-q}$ are the current and the q previous values of a random process, usually emanating from a normal (Gaussian) distribution with mean zero and $\varphi_1 \ldots \varphi_p, \psi_1 \ldots \psi_q$ are the model parameters.

The ARMA(p,q)-based predictor approximates the real process value x_t by a predicted value $\hat{x}_t$, computed as

$$\hat{x}_t = \varphi_1 x_{t-1} + \ldots + \varphi_p x_{t-p} + \psi_1 a_{t-1} + \ldots + \psi_q a_{t-q}, \tag{10.31}$$

The residual a_{t-i} represents the error between the real process value x_{t-i} and the predicted value $\hat{x}_{t-i}$.

The AR(p) and MA(q) models are special cases of the ARMA(p,q) model, where AR(p) is described as

$$x_t = \varphi_1 x_{t-1} + \varphi_2 x_{t-2} + \ldots + \varphi_p x_{t-p} + a_t, \tag{10.32}$$

and MA(q) is described as

$$x_t = a_t + \psi_1 a_{t-1} + \psi_2 a_{t-2} + \ldots + \psi_q a_{t-q}. \tag{10.33}$$

ARMA modeling is very fast, but of limited applicability due to strong modeling assumptions (e.g., stationary process, linear interdependencies, Gaussian noise).

A natural, less restrictive, generalization of the linear ARMA and AR models to the nonlinear cases leads to the NARMA model

$$x_t = h(x_{t-1}, x_{t-2}, \ldots, x_{t-p}, a_{t-1}, \ldots, a_{t-q}) + a_t, \tag{10.34}$$

and the NAR model

$$x_t = h(x_{t-1}, x_{t-2}, \ldots, x_{t-p}) + a_t, \tag{10.35}$$

where h is an unknown smooth function.

The AR-, MA-, NARMA- and NAR-based predictors are obtained from their corresponding models analogous to obtaining the ARMA-based predictor (equation (10.31)) from the ARMA model (equation (10.30)). However, the NARMA and NAR models are very complex, thus difficult to use in real life applications. Fortunately, they are closely related to more practical nonlinear models, the neural networks. Recurrent and feedforward neural networks have been proposed in (Connor et al., 1994; Werbos, 1992) for simulating NARMA and NAR models respectively. An invertible (Box et al., 1994) NARMA-based predictor can be approximated as

$$\begin{aligned} \hat{x}_t &= h(x_{t-1}, \ldots, x_{t-p}, a_{t-1}, \ldots, a_{t-q}) \\ &\approx \sum_{i=1}^{m} W_i f\Big(\sum_{j=1}^{p} w_{ij} x_{t-j} + \sum_{j=1}^{q} w'_{ij}(x_{t-j} - \hat{x}_{t-j}) + \theta_i\Big) + \Gamma, \end{aligned} \tag{10.36}$$

where f represents a nonlinear, smooth and bounded function and $a_k = x_k - \hat{x}_k$, for all $k \in \{t-q, \ldots, t-1\}$. This approximation of the NARMA-based model corresponds to the recurrent neural network from Fig. 10.2, in which w_{ij} are the weights between external inputs and hidden neurons, w'_{ij} are the weights between context inputs and hidden neurons, W_i are the weights between hidden and output neurons, θ_i are the hidden neuron biases, Γ is the output neuron bias and f is the

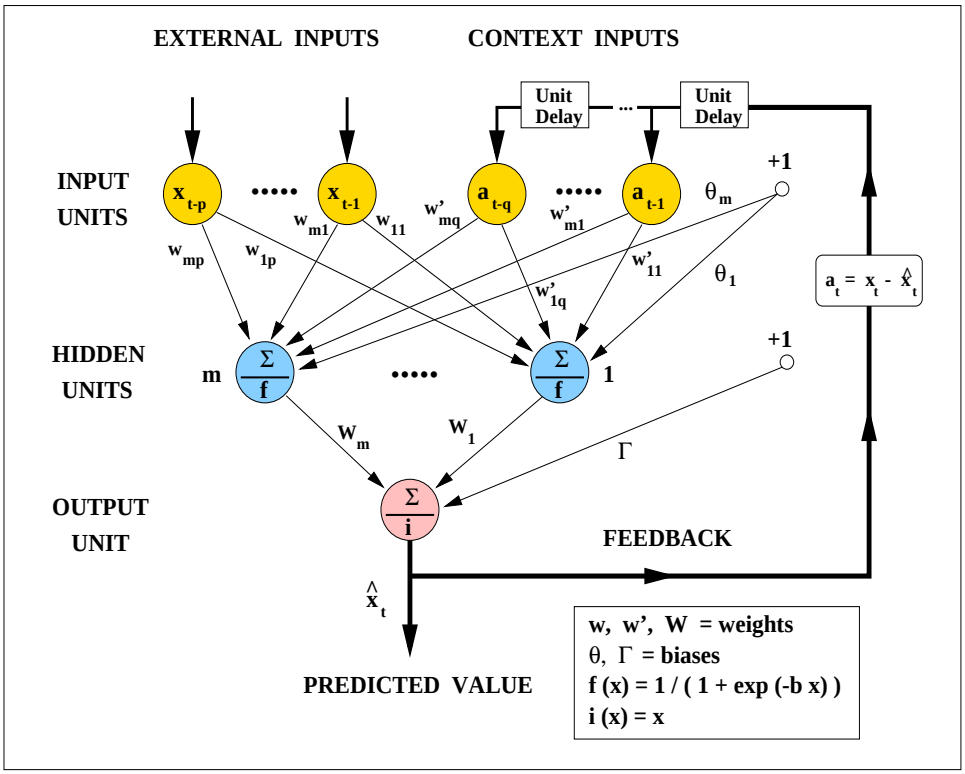

Figure 10.2 Stochastic Model Approximation

activation function of the hidden neurons. Similarly, a NAR-based predictor can be approximated as

$$\hat{x}_t = h(x_{t-1}, \ldots, x_{t-p}) \approx \sum_{i=1}^{m} W_i f(\sum_{j=1}^{p} w_{ij} x_{t-j} + \theta_i) + \Gamma, \tag{10.37}$$

obtained by disconnecting the context inputs $a_{t-1} \ldots a_{t-q}$ in Fig. 10.2. The parameters w_{ij}, w'_{ij}, W_i, θ_i and Γ can be estimated from examples by gradient descent optimization (Werbos, 1994).

A special case of a particular interest in this study is the approximation of a linear AR model by a feedforward neural network. Although the neural network from Fig. 10.2 in which the hidden layer and the feedback connections are removed is computationally equivalent to the linear AR model, such a trivial network is of no interest since it is not able to perform better than the equivalent AR model. More interesting (see Appendix C) is the approximation of an AR(p) model with parameters $\varphi_1, \ldots, \varphi_p$ by a neural network with p inputs, p hidden units and

interconnection parameters

$$\begin{aligned} w_{ij} &= \delta_{ij} \\ \theta_i &= 0 \\ W_i &= \tfrac{4}{\beta}\varphi_i \\ \Gamma &= -\tfrac{2}{\beta}\sum_{i=1}^{p}\varphi_i, \end{aligned} \tag{10.38}$$

for all $i, j \in \{1, \ldots, p\}$.

In (Drossu and Obradovic, 1996a) it was shown that the neural network weight initialization based on AR model parameters could significantly shorten the neural network training process by providing an initial position on the error surface which is closer to the minimum as compared to a randomly chosen position. In addition, stochastic analysis could provide some initial knowledge regarding appropriate neural network architecture and data sampling rate. The attempt of using linear stochastic analysis prior knowledge is supported by the fact that "many non-linear systems can be described fairly well by linear models and for such systems it is a good idea to use insights from the best linear model to select the regressors for the neural network model" (Sjoberg et al., 1994, 1995). The objective of the approach proposed in (Drossu and Obradovic, 1996a) is not to obtain "the optimal" neural network architecture for a given problem, but to provide rapidly an architecture with close to optimal performance. Since information is obtained from a linear model, for more complex problems the neural network might be over-dimensioned (similar performance could be obtained using a smaller machine and less learning examples). However, the exhaustive trial and error procedure involved for determining such an optimal machine could be costlier than the stochastic analysis based alternative.

10.2.4 An Illustrative Example

Our experiments performed in (Drossu and Obradovic, 1996a) tested whether the most appropriate linear stochastic model can provide an indication of the appropriate number of neural network inputs. Additionally, they explored whether initial neural network weights obtained from the stochastic model as described by equations (10.38) are appropriate. In the case of larger prediction horizons, the experiments also analyzed whether an adequate data sampling rate could be obtained from stochastic modeling.

All experiments encompassed a pre-processing, consisting of both a logarithmic smoothing and a first order differentiation for stationarization purposes and the neural network weight optimization was performed using gradient descent. The validity of the stochastic modeling prior knowledge for selecting an adequate neural network architecture, initial weights and sampling rate was tested in the context of two very different data sets:

- Mackey-Glass data.

The first data set is a deterministic time series, also known as Mackey-Glass series,

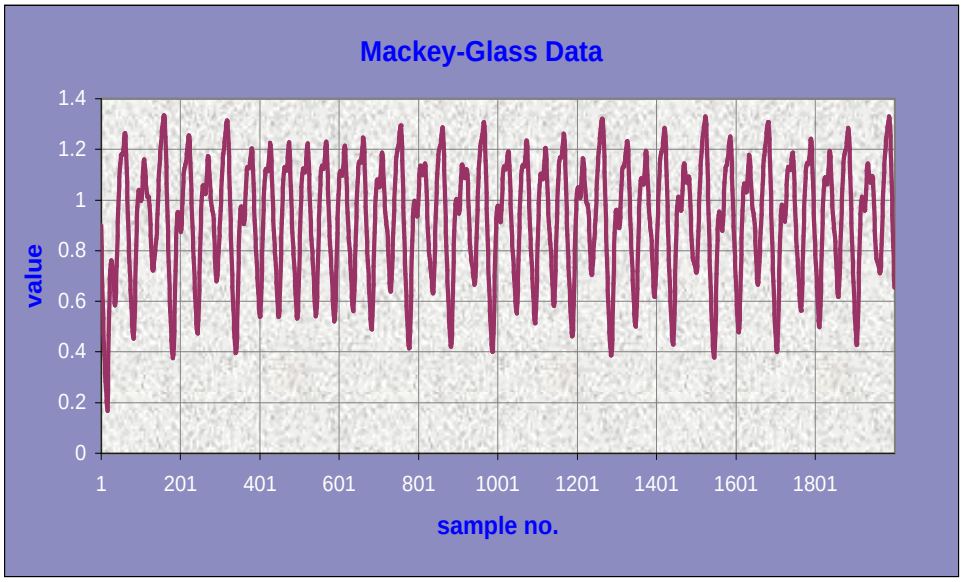

Figure 10.3 Mackey-Glass Data

obtained by integrating the delay differential equation,

$$\frac{dx(t)}{dt} = \frac{Ax(t-\tau)}{1+x^{10}(t-\tau)} - Bx(t)$$

Experiments were performed for $A = 0.2$, $B = 0.1$, $\tau = 17$, case in which the system exhibits chaotic behavior. The difficulty associated with this data set is the high *nonlinearity*. The data set consisted of 12000 samples, the first 2000 shown in Fig. 10.3. The time series appears to be quasi-periodic with fairly long smooth segments. This suggests that the prediction of most of the series (except for the turning points, possibly) should be fairly easy for an adequate predictor. In accordance to previously published results (Lapedes and Farber, 1987), a sampling rate six was used for predicting 6 or 6*k steps ahead. Hence, the original data set was sampled at a rate 6 to generate a new 2000 samples data set which was used for experimentation on prediction horizons 1 and k. The first 1000 samples of this "filtered" data were used for training, whereas the last 1000 samples were used for testing.

- Entertainment video traffic data.

The second data set used in the experiments consisted of a real life, compressed, entertainment video traffic data used in an ATM (Asynchronous Transfer Mode) network, in which each sample represents the size of a corresponding compressed video frame (Drossu et al., 1995). The characteristics of this data set are *non-stationarity* (data distribution changes over time) and as the existence of "outliers"

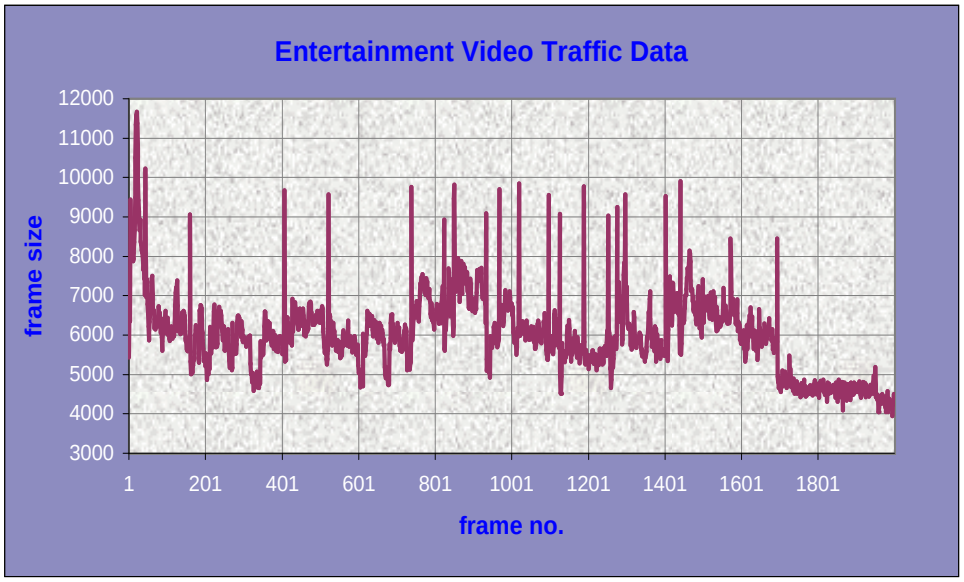

Figure 10.4 Entertainment Video Traffic Data

(values very different from neighboring ones). The problem is especially difficult since the outliers contain useful information that cannot be discarded through filtering. Hence, it is not sufficient to be able to accurately predict the (easily predictable) smooth sections of the time series, but also the outliers. The data set considered in our experiments consisted of 2000 samples (shown in Fig. 10.4).

10.2.4.1 Predicting the Near Future

In these experiments, the neural network predictors attempted to predict one step ahead of time using the Mackey-Glass time series. The neural network weights were initialized either with small random values, or from the corresponding AR parameters as in equations (10.38). The results presented for the neural networks initialized with random weights were averaged over 10 runs.

The results were compared versus an earlier reported "optimal" neural network topology with 4 inputs and two hidden layers of 10 units each (Lapedes and Farber, 1987), in which the number of inputs was determined based on dynamical system analysis and applying Takens' theorem, as discussed in Section II.B, while the number of hidden layers and hidden units was determined through extensive experimentation on a supercomputer. The predictors' accuracy was evaluated according to the coefficient of determination (Anderson-Sprecher, 1994), computed

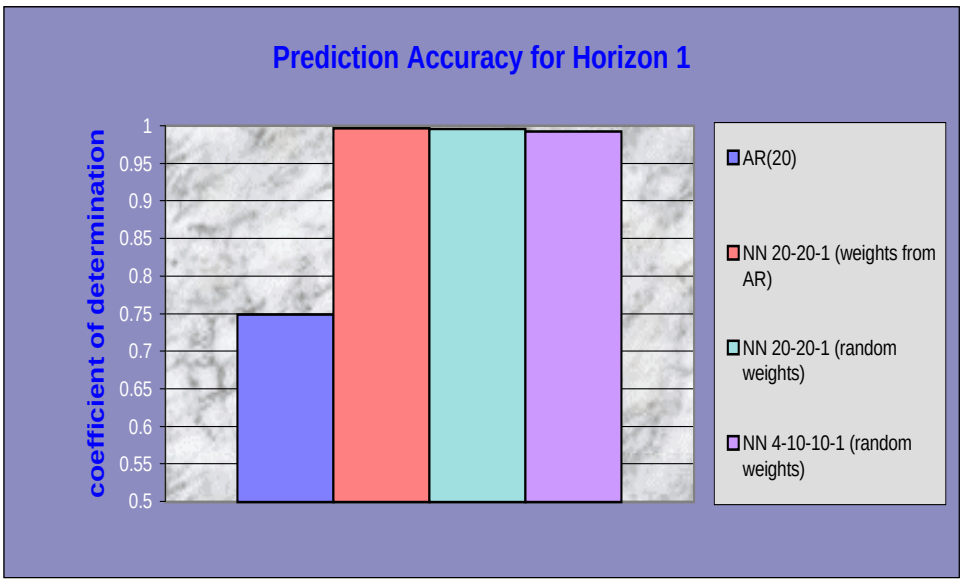

Figure 10.5 Prediction Accuracy for Horizon 1 on Mackey-Glass Data

as

$$r^2 = 1 - \frac{\sum_{t=1}^{N}(x_t - \hat{x}_t)^2}{\sum_{t=1}^{N}(x_t - \bar{x})^2}, \tag{10.39}$$

where N represents the number of samples, x_t and $\hat{x}_t$ denote the actual and the predicted process values, respectively, while $\bar{x}$ denotes the mean of the actual data. For a perfect predictor, the coefficient of determination should be 1, whereas for a trivial mean predictor (one whose every prediction equals the mean of the actual data), the coefficient of determination is 0.

The r^2 values, summarized in Fig. 10.5, indicated an AR(20) model as the most appropriate linear model, thus suggesting the use of a feedforward neural network with 20 inputs. Varying the hidden layer size suggested that a number of hidden units equal to the number of inputs was an appropriate choice, thus allowing also the neural network weight initialization using AR model parameters. Whether starting from random weights or initializing the weights from the AR parameters, the neural networks yielded a very similar prediction accuracy. Starting the neural network learning process with weights initialized from the AR parameters could, nevertheless, offer the benefit of being close to a minimum of the error surface, hence shortening the learning process. It would also eliminate the necessity of running a number of experiments in which the weights are initialized with different random values in order to obtain an averaged performance. On the other hand, to avoid the

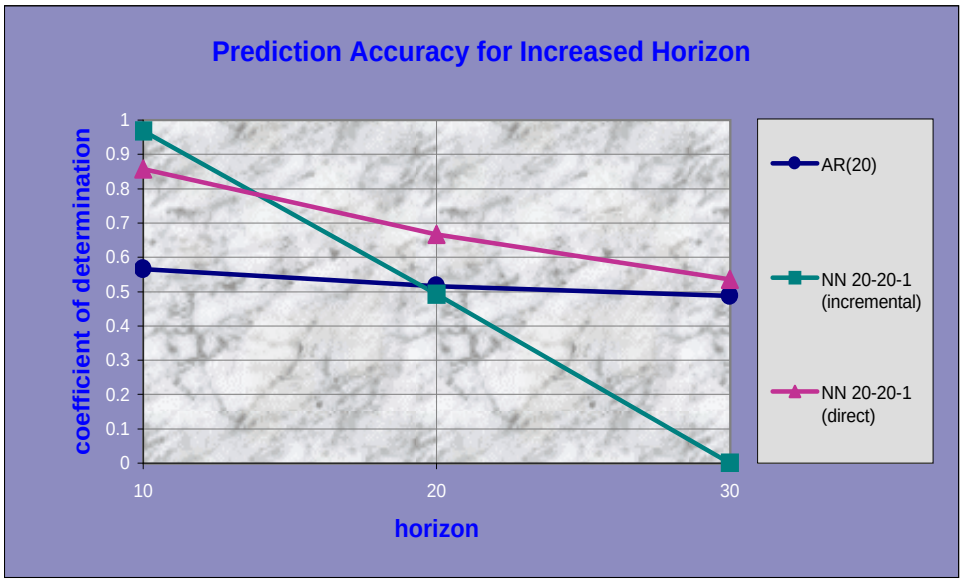

Figure 10.6 Coefficient of Determination for Increased Prediction Horizon on Mackey-Glass Data

"freezing" of the learning process in a local minima a small additive noise to the initial weight values could be desirable.

The conclusions drawn from this experiment are (Drossu and Obradovic, 1996a):

- The performance of the neural networks was much better as compared to the most appropriate stochastic model, this being consistent with the nonlinearity of the time series.
- The neural network based on stochastic prior knowledge (both regarding the number of inputs and appropriate initial weight values) performed similar to the "optimal" neural network architecture, supporting the stochastic information based design approach. This is a very useful finding, since it confirms that useful information can be extracted from a linear model even in the case in which the underlying time series is highly nonlinear.

10.2.4.2 Predicting further into the Future

For horizon h larger than one, the prediction can be done either in a *direct* or in an *incremental* fashion. In the direct approach, the neural network is trained to predict directly the h-th step ahead without predicting any of the intermediate $1, \ldots, h-1$ steps. In the incremental approach, the neural network predicts all the intermediate values up to h steps ahead by using the previously predicted values as inputs when

predicting the next value. The experiments, performed also on the Mackey-Glass series, were concerned with the decrease in prediction accuracy when significantly increasing the prediction horizon (Drossu and Obradovic, 1996a). For this purpose the AR(20) and the NN 20-20-1, trained as in the experiment for prediction horizon 1, were used to incrementally predict the process values up to 30 steps ahead (this corresponds to 180 steps ahead in the "unfiltered" series presented in Fig. 10.3). In the case of the neural network, the weights were initialized from the AR parameters. The values for the coefficient of determination resulting from these experiments are presented in Fig. 10.6.

The results indicated that the performance of the neural network was much better when predicting the near future, but it decreased dramatically and after about 30 steps ahead the predictor became completely unusable. This was an indication of the instability of the trained neural network (an undesirable error accumulation when using the incremental approach). For this reason, three 20-20-1 neural networks were trained in the direct fashion for predicting 10, 20 and 30 steps ahead, respectively. The values for their coefficients of determination, obtained as averages over 10 runs with different initial random weights, are also included in Fig. 10.6 and they are significantly better than the corresponding ones for the AR(20) model. However, for prediction horizon 10 the coefficient of determination for the direct approach was worse than for the incremental approach.

The conclusion drawn from this experiment is (Drossu and Obradovic, 1996a):

• Although an incremental approach for neural network training in the case of an increased prediction horizon has the advantage of training a single neural network and using it afterwards for predicting as many steps ahead as desired, the system can be unstable resulting in a dramatic error accumulation when increasing the prediction horizon. For this reason for larger prediction horizons it is desirable to analyze both the incremental and the direct training approach and to select the more appropriate one for each particular prediction horizon.

10.2.4.3 Selecting the Sampling Rate

For a larger prediction horizon different sampling rates can be employed, making the trial and error neural network architecture selection even more impractical. Consequently, in this experiment the choice of an appropriate sampling rate based on the stochastic modeling prior knowledge was explored. In addition, it was also tested whether an appropriate AR(p) model indicated the use of a feedforward neural network with p external inputs whose initial weights could be set according to the AR parameters.

The entertainment video traffic data was used for experimentation for a prediction horizon 10 (the 10th step ahead process value is predicted) (Drossu and Obradovic, 1996a). To predict the process at time step $t + 10$ using k process values up to time t, the following uniform sampling rates (divisors of the prediction horizon) were considered:

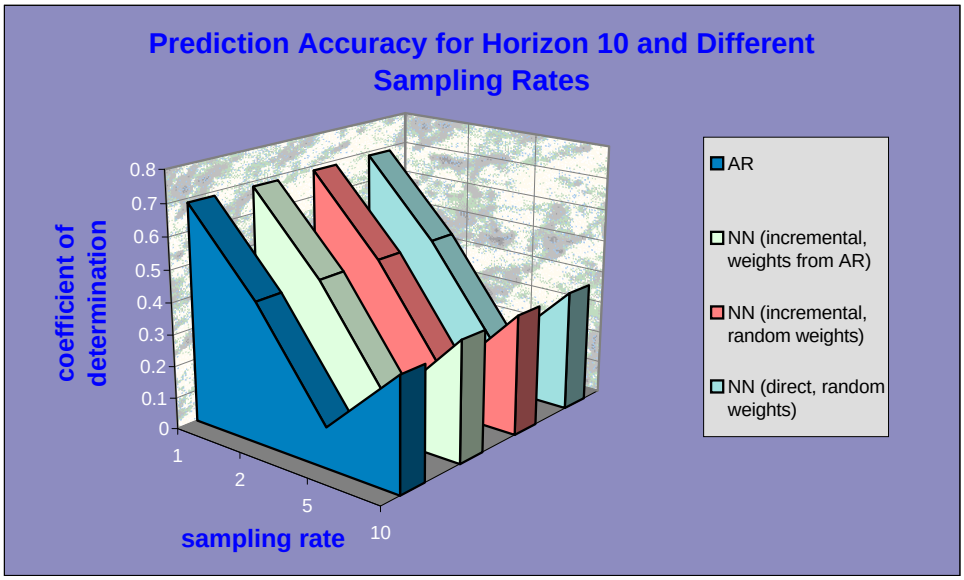

Figure 10.7 Prediction Accuracy for Horizon 10 and Different Sampling Rates on Entertainment Video Traffic Data

- sampling rate 1, where the k previous process values are $x(t), x(t-1), x(t-2), \ldots, x(t-k+1)$;

- sampling rate 2, where the k previous process values are $x(t), x(t-2), x(t-4), \ldots, x(t-2*(k-1))$;

- sampling rate 5, where the k previous process values are $x(t), x(t-5), x(t-10), \ldots, x(t-5*(k-1))$;

- sampling rate 10, where the k previous process values are $x(t), x(t-10), x(t-20), \ldots, x(t-10*(k-1))$.

All neural network results were obtained either by initializing the weights from the AR parameters, or averaged over 10 runs with different initial random weights.

The coefficient of determination for the most appropriate AR models obtained for different sampling rates, as well as for the corresponding neural network models are presented in Fig. 10.7. The stochastic models indicated a sampling rate 1 as the most appropriate confirmed also by their neural network counterparts. It could also be observed that the performance of the neural networks with weights initialized according to the AR parameters was very similar to that of the neural networks averaged over 10 runs with different initial random weights.

The results obtained for the most appropriate stochastic model, as well as for different representative neural networks when using a sampling rate 1 indicated that the neural network having a number of inputs equal to the order of the most

appropriate AR model yielded the best prediction.

The conclusions that could be drawn from these experiments are (Drossu and Obradovic, 1996a):

• The data sampling rate indicated by the stochastic models seems to be appropriate also for the neural network models.

• The prior knowledge provided by the stochastic analysis regarding the number of external inputs, and appropriate initial weight values is effective also for larger horizons.

• The performance of the AR models and the corresponding neural networks is comparable, this indicating a linearity of the problem under consideration.

10.3 Indirect Information Extraction Procedures

10.3.1 Knowledge of Properties of the Target Function.

Whenever a prediction model, whether neural network or other, is trained on a data set, the only information that the model can extract is from the data itself. In many real-life applications, however, some properties of the function to be approximated are known ahead of time. The use of these properties, called *hints* (Abu-Mostafa, 1995b,a), is of major importance especially in problems with scarce (or costly to obtain) or noisy data, like financial forecasting problems, in which hints can improve the model's accuracy drastically. Nevertheless, a non-valid hint can deteriorate the performance of the model considerably (Abu-Mostafa, 1995b), so care must be taken in order to analyze the validity of hints. Hints play an important role in improving the generalization ability (predictive accuracy) of the model by imposing constraints on the target function which has to be learned. This would correspond to restricting the search space for valid target functions by eliminating those which could potentially fit the noise instead of focusing on the relevant information contained in the data.

Two modalities for incorporating hints in the neural network learning process we proposed in (Abu-Mostafa, 1995b,a):

- creating additional "virtual" training examples;
- imposing constraints on the learning process by modifying the cost function.

The two modalities of embedding hints into the learning process will be illustrated in the context of two examples presented in (Abu-Mostafa, 1995a). The first one deals with the case in which the target function to be approximated is known to be odd. In this case, if (x, y) is known to be a valid training example, a virtual example $(-x, -y)$ can be created which could provide an additional ready-to-use training example (if not already present in the training set). On the other hand, learning the oddness property of the target function can be enforced during the learning

process (e.g., using gradient descent optimization). Similar to learning a function by minimizing the squared error between desired and real neural network output, $(y(x)-\hat{y}(x))^2$, the oddness property can be enforced by minimizing $(\hat{y}(x)+\hat{y}(-x))^2$. This requires the input of both x and $-x$ to the network and minimizing the difference between the two outputs. A second example of incorporating hints assumes the target function to be invariant to certain transformations (e.g., scaling, translation, rotation in pattern recognition problems). Virtual examples can be produced by considering an available training example (x, y) and creating the virtual example (x', y), in which x' represents the value obtained from x by applying the invariance transformation. The invariance property can also be imposed during learning by minimizing, in addition to the squared error sum, a sum of terms of the form $(\hat{y}(x) - \hat{y}(x'))^2$. Many other additional hints like symmetry, monotonicity, etc., can also be easily incorporated into the neural network learning.

Although not an actual way of accelerating the neural network design process, the use of known properties of the target function is an easy and cost-effective way of improving a time series predictor's accuracy, which can be used in conjunction with any other direct method presented earlier.

10.3.2 Non-stationarity Detection

A *non-stationary* time series can be described as a time series whose "characteristic parameters" change over time. Different measures of stationarity can be employed to decide whether a process is stationary or not (Papoulis, 1984). In practice, confirming that a given time series is stationary is a very difficult task unless a closed-form expression of the underlying time series is known, which is rarely the case. On the other hand, non-stationarity detection can be reduced to identifying two sufficiently long, distinct data segments that have significantly different statistics (distributions). In practice, common tests for comparing whether two distributions are different are (Press et al., 1992):

- Student's t-test;
- F-test;
- chi-square test;
- Kolmogorov-Smirnov test.

Student's t-test is applied to identify the statistical significance of a difference in means of two distributions assumed to have the same variance, whereas the *F-test* evaluates the statistical significance of a difference in variances. More commonly, if there aren't any assumptions regarding the means or variances of the distributions, a chi-square or a Kolmogorov-Smirnov test, summarized in Appendix D, are performed.

If time is not an issue, non-stationary time series prediction can be accomplished by performing on-line learning using a *sliding window* technique (Chenoweth and Obradovic, 1996), in which a new prediction model is built whenever a new data

sample becomes available. However, in many real-life problems the data arrival rate is high, which makes this approach completely infeasible due to the computational complexity involved in repeatedly building neural network prediction models. An alternative encountered in practice is the *uniform retraining* technique, in which an existing neural network prediction model is used for a pre-specified number of prediction steps (which we call *reliable prediction interval*), followed by the replacement of the existing model by one constructed using more recent data. A major disadvantage of uniform retraining is that it is often hard to determine an appropriate reliable prediction interval, as it might be changing over time.

Although theoretically possible, in practice it might be very difficult to efficiently learn a single global neural network model for a non-stationary time series prediction. An obvious difficulty of such a global approach is the selection of neural network modeling parameters that are appropriate for all data segments. Additional serious problems include different noise levels in various data segments resulting in local overfitting and underfitting conflicts (it would be desired to stop training as not to overfit some data segments, while other data segments would still require additional training).

An interesting multi-model attempt to predict *piecewise stationary* time series, where the process switches between different regimes, is by using a *gating network*, in which a number of neural network experts having an identical structure are trained in parallel, and their responses are integrated by another neural network trained simultaneously with the expert networks (Weigend et al., 1995). Briefly, due to an adequate combination of activation and error functions that encourages localization, in a gating network each expert network tends to learn only a subset of the training data, thus devoting itself solely to a sub-region of the input space. This competitive integration method showed quite promising results when predicting a non-stationary time series having two regimes, but is not likely to extend well to more complex non-stationary processes due to overfitting problems of training a gating network system consisting of too many expert networks. In addition, the time required to train a complex gating network is likely to be prohibitively long for many real-life time series prediction problems.

In (Drossu and Obradovic, 1996b) we proposed three different time series prediction scenarios which depend on the amount of prior knowledge regarding a potential data distribution:

1. switching among historically successful neural network models (SWITCH);
2. reusing one of historically successful available neural network models, or designing a new one (REUSE);
3. retraining a neural network model when signaled, without relying on any historically successful model (RETRAIN).

The SWITCH scenario assumes a piecewise stationary, multi-regime time series and a library containing models for all regimes. To simplify the presentation we will assume two regimes and their associated historically successful models. The

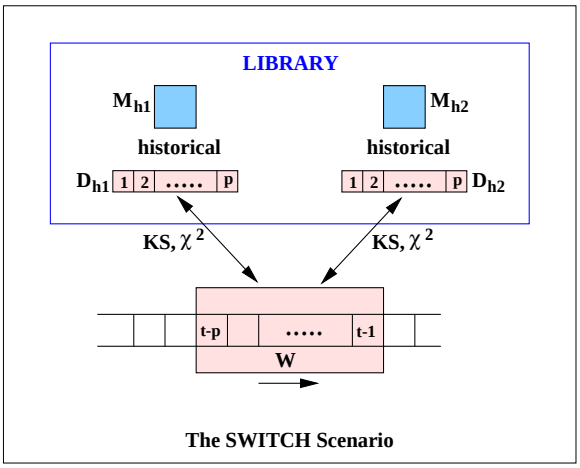

Figure 10.8 Statistics-based SWITCH

objective is to detect in real-time which of the two models to use for prediction at any given time step. The REUSE scenario assumes the potential existence of a repetitive regime along with an associated library model. The objective is to decide in real-time whether to use the existing previously successful historical model for prediction, or to retrain a new neural network on current data. Finally, the RETRAIN scenario is not assuming any prior knowledge regarding the non-stationarity type. The objective is to decide in real-time when to discard a neural network predictor and retrain a new one on current data. The SWITCH and the REUSE scenarios are proposed in order to efficiently forecast piecewise stationary processes with full or partial understanding of the number of different regimes, while the RETRAIN scenario is proposed for forecasting completely unknown higher order non-stationary processes.

The three scenarios can be used in the context of statistics- and accuracy-based distribution-change signaling techniques, discussed as follows.

10.3.2.1 Statistics-Based Signaling

This signaling technique attempts to identify changes in the data distribution by comparing the similarity of different data segments using either the chi-square or the K-S statistics.

For the SWITCH scenario (see Fig. 10.8), two historical data segments, D_{h1} and D_{h2}, both of length p, along with their neural network models, M_{h1} and M_{h2}, trained on these segments are kept in a library. A current window, W, containing the p latest available data is compared in distribution (using either the chi-square or the K-S tests) to D_{h1} and D_{h2}, in order to decide which of the two historical data segments is more similar to it. The library model corresponding to the more appropriate historical data segment is then used for predicting the next time series value.

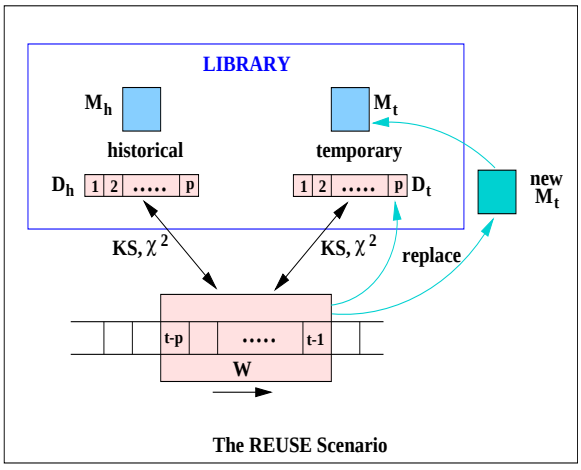

Figure 10.9 Statistics-based REUSE

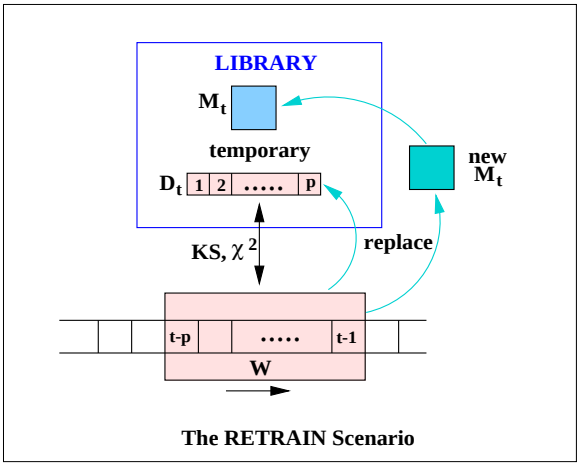

Figure 10.10 Statistics-based RETRAIN

For the REUSE scenario (see Fig. 10.9), a single historical data segment, D_h, used to build a previously successful neural network model, M_h, as well as a temporary data segment, D_t, used to build a temporary neural network model, M_t, both of length p, are kept in a library. The models M_h and M_t are also stored in the library. A current window, W, containing the p latest available data is compared in distribution to D_h and D_t, in order to decide whether to continue using one of the library models or to train a new model. For this purpose, a threshold has to be imposed on the confidence value obtained from the chi-square or K-S tests. If the test indicates more confidence in M_h, provided that the confidence value for M_h is larger than the specified threshold, then M_h is used for the current prediction. Similarly, if we are more confident in M_t and the confidence value is larger than the threshold, then M_t is used for the current prediction. Otherwise (none of the

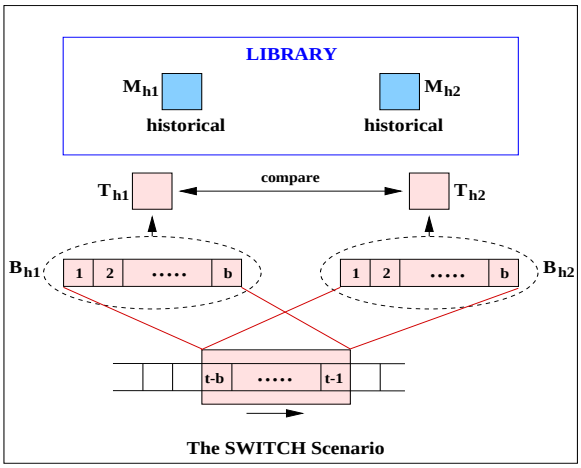

Figure 10.11 Accuracy-based SWITCH

confidence values is larger than the imposed threshold), a new temporary neural network model is trained on W and it replaces M_t, whereas W replaces D_t in the library. The new model is then used for the current prediction.

In the case of the RETRAIN scenario (see Fig. 10.10), a data segment, D_t, of length p used to build a temporary neural network model, M_t, is stored in a library. A current window, W, containing the p latest available data is compared in distribution to D_t, in order to decide whether to continue using M_t, or discard it and train a new neural network model. Once again, a threshold has to be imposed on the confidence value obtained from the chi-square or K-S tests in order to decide when the current model becomes inappropriate. If M_t is considered to be inadequate, W replaces D_t and a new neural network model trained on W replaces M_t in the library, which is used for the current prediction.

10.3.2.2 Accuracy-Based Signaling

The objective of this signaling technique, also proposed in (Drossu and Obradovic, 1996b) is to identify data distribution changes by measuring recent prediction accuracies of previously successful models.

For the SWITCH scenario (see Fig. 10.11), two historically successful neural network models, M_{h1} and M_{h2}, are kept in a library. At each time step, the two models are compared based on their accuracy measured on a buffer containing b most recent process values, and the more accurate model is used for the current prediction.

For the REUSE scenario (see Fig. 10.12), a historically successful neural network model, M_h, as well as a temporary neural network model, M_t, are kept in a library. Similar to the SWITCH scenario, the accuracy of the two models is compared on the b most recent process values. The model having a better accuracy is used for predicting the current step, unless none of the models is a sufficiently good predictor

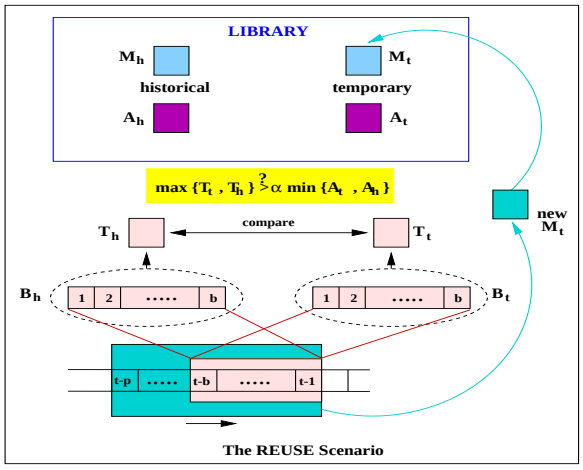

Figure 10.12 Accuracy-based REUSE

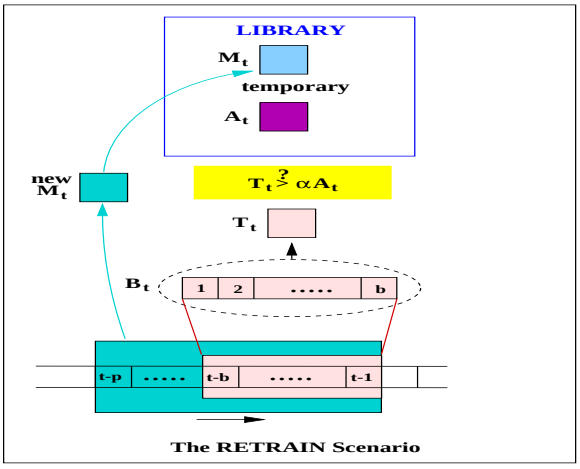

Figure 10.13 Accuracy-based RETRAIN

on the b most recent process values. A model is considered to be sufficiently good if its accuracy on the b most recent process values is above $\alpha \min\{A_h, A_t\}$, where α is a pre-specified threshold in the (0,1) range, while A_h and A_t are the training accuracies for the historical and the temporary model, respectively, computed on the process values used to build them. If none of the two existing models is satisfactory, a new neural network model is trained, that replaces M_t in the library and is also used for the current prediction.

In the case of the RETRAIN scenario (see Fig. 10.13), a temporary neural network model, M_t, is stored in a library. Additionally, a corresponding training accuracy, A_t, is measured as for the REUSE scenario. If the accuracy M_t, measured on the b most recent process values is αA_t, model M_t is used for the current prediction. Otherwise, a new neural network model is trained which replaces M_t in

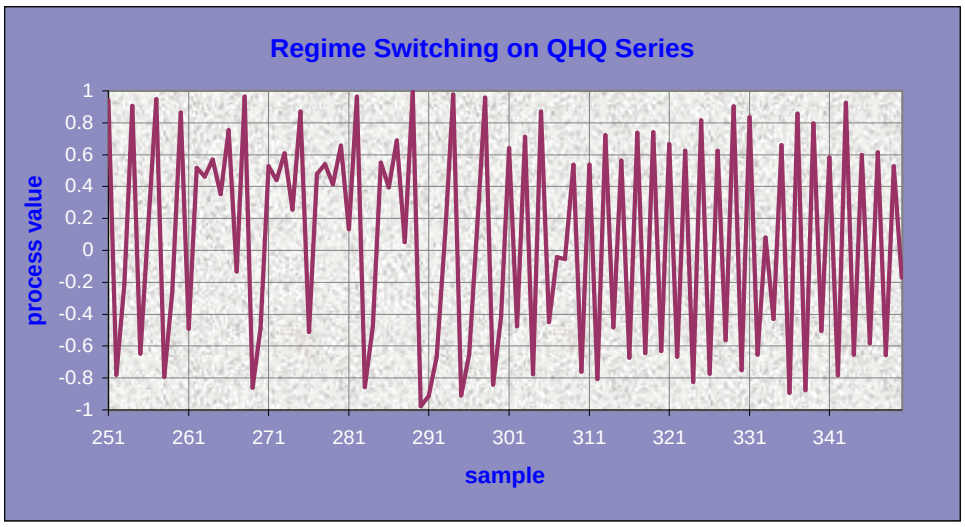

Figure 10.14 Regime Switching on QHQ Series

the library and is also used for the current prediction.

10.3.3 An Illustrative Example

In (Drossu and Obradovic, 1996b), non-stationary time series prediction experiments were performed on generic data which allow a rigorous control of regime switching between distributions, as well as the possibility of computing the performance of an optimal predictor.

The time series used there were constructed by mixing data stemming from a deterministic chaotic process (Q) and a noisy, non-chaotic process (H) used earlier in (Weigend et al., 1995). The processes Q and H were generated according to the following rules:

$$x_{t+1} = 2(1 - x_t^2) - 1 \quad (Q)$$

$$x_{t+1} = \tanh(-1.2x_t + \epsilon_{t+1}) \quad (H),$$

where $\{\epsilon_t\}$ is a white noise process with mean 0 and standard deviation 0.32.

A first time series, denoted by QHQ, was created by concatenating three data sections of lengths 300, 400, and 500 samples, respectively, in which the first and the last data segments stemmed from the Q process, whereas the second data segment stemmed from the H process.

A segment of the QHQ time series comprising the first regime switch from process Q to process H (time series data samples 251-350) is presented in Fig. 10.14.

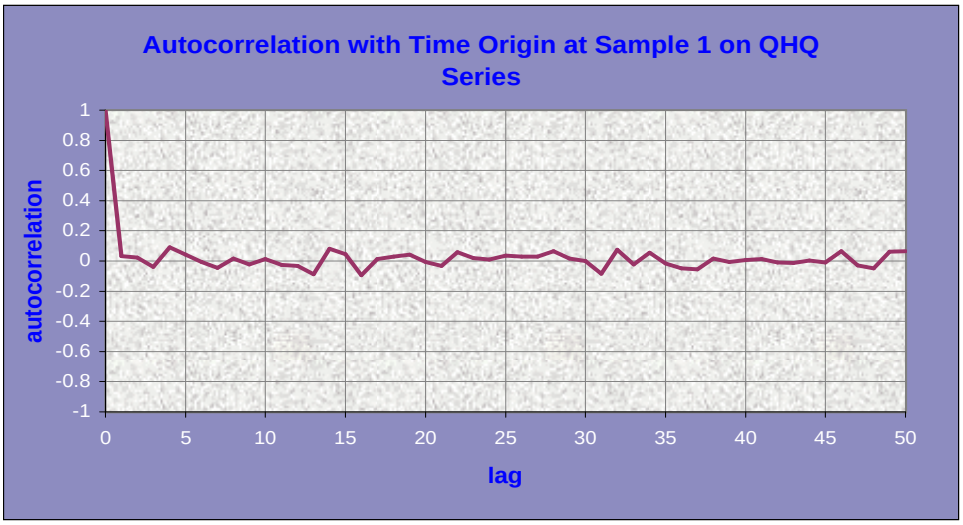

Figure 10.15 Autocorrelation with Time Origin at Sample 1 on QHQ Series

Although the Q and H processes have basically the same means and variances, as well as data ranges, Fig. 10.14 illustrates the different time behavior of the two processes. Indeed, the autocorrelation plots for lags up to 50 on the first 300 and the next 300 time series data samples, shown in Figs. 10.15 and 10.16, indicate a dependence of autocorrelation on time origin, meaning that the underlying mixed time series is not wide-sense stationary (Papoulis, 1984).

Two feedforward neural networks having 2 input units, two hidden layers of 4 units each and 1 output unit were trained (using the gradient descent algorithm) on two data segments stemming from the Q and the H processes, respectively.

10.3.3.1 The SWITCH Scenario

The experiments compared statistics-based signaling and accuracy-based signaling to a single-model predictor as well as to an optimal predictor (see Fig. 10.17). The single-model predictor is a library model used for predicting the entire time series, whereas the optimal predictor is obtained by using both library models and assuming that the switching points between distributions are detected without any delay (this is infeasible in practice unless the regime switching rules are entirely understood).

Although the statistics-based signaling technique yields a significantly better prediction accuracy as compared to the single-model predictor, the results show the drastic superiority of accuracy-based signaling, which provides excellent results for buffer sizes over a fairly wide range, 2-30. It could also observed that these buffer sizes lead to performance which is comparable to that achieved when the regime

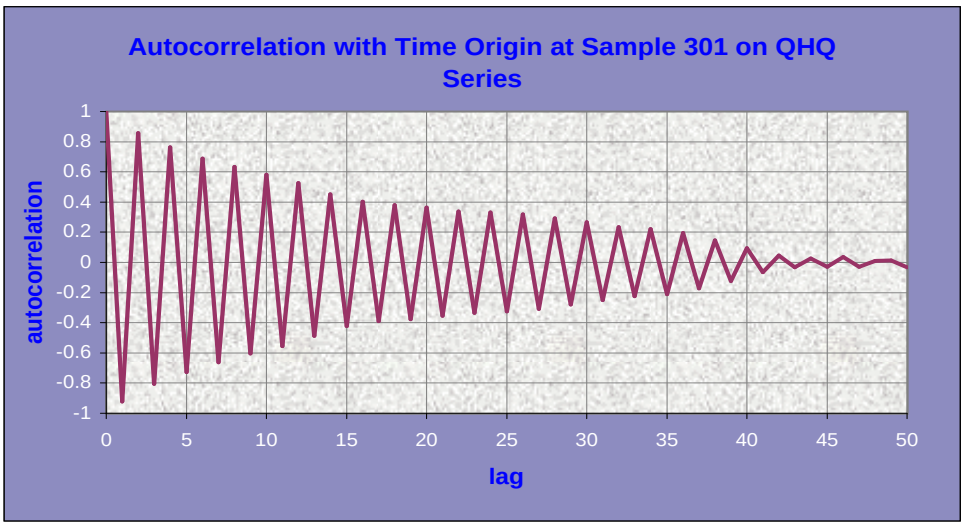

Figure 10.16 Autocorrelation with Time Origin at Sample 301 on QHQ Series

switching points are completely known (optimal predictor curve).

10.3.3.2 The REUSE Scenario

The results obtained in (Drossu and Obradovic, 1996b) using the accuracy-based signaling technique for buffer sizes 50 and 100, averaged over ten runs, are shown as the first two bars in Fig. 10.18. The bars represent the 99% confidence regions for the r^2 means, based on *Student's t* distribution with 9 degrees of freedom. In all experiments the mean values for the coefficient of determination were significantly better than those obtained by the SWITCH scenario with statistics-based signaling, with small deviations given by the 99% confidence regions. Consequently, results obtained using the statistics-based signaling were not reported for the REUSE scenario. On the other hand, although the averaged value of the coefficient of determination was larger for all experiments using a shorter buffer, a statistically significant difference could not be claimed (the 99% confidence regions overlap). The number of neural network retrainings in the experiments with buffer length 100 varied between 3 and 7, whereas it varied between 5 and 14 in the case of buffer length 50. These figures indicate that the experiments on longer buffer are computationally more efficient. However, even for the shorter buffer, the number of retrainings is very small as compared to the total number of predictions.

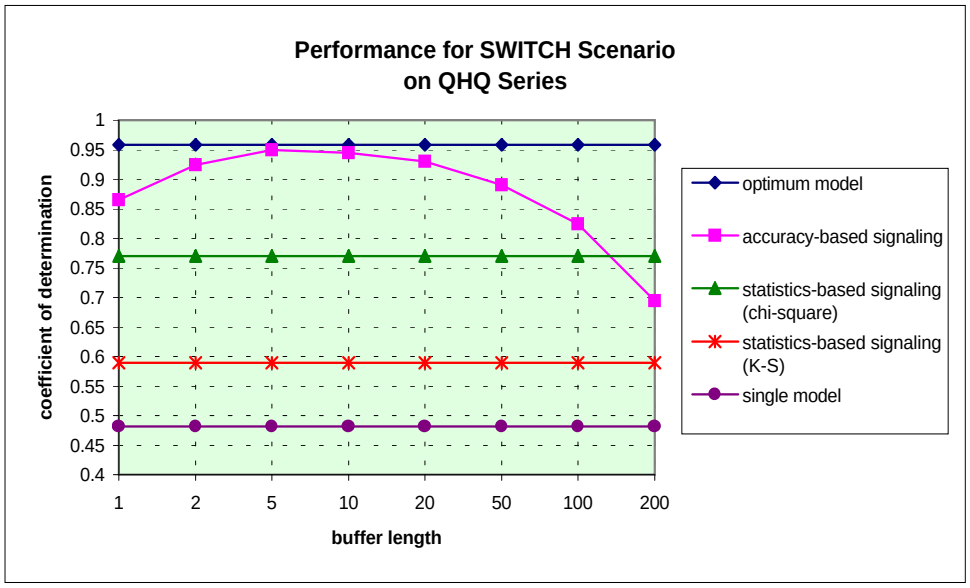

Figure 10.17 Performance for SWITCH Scenario on QHQ Series

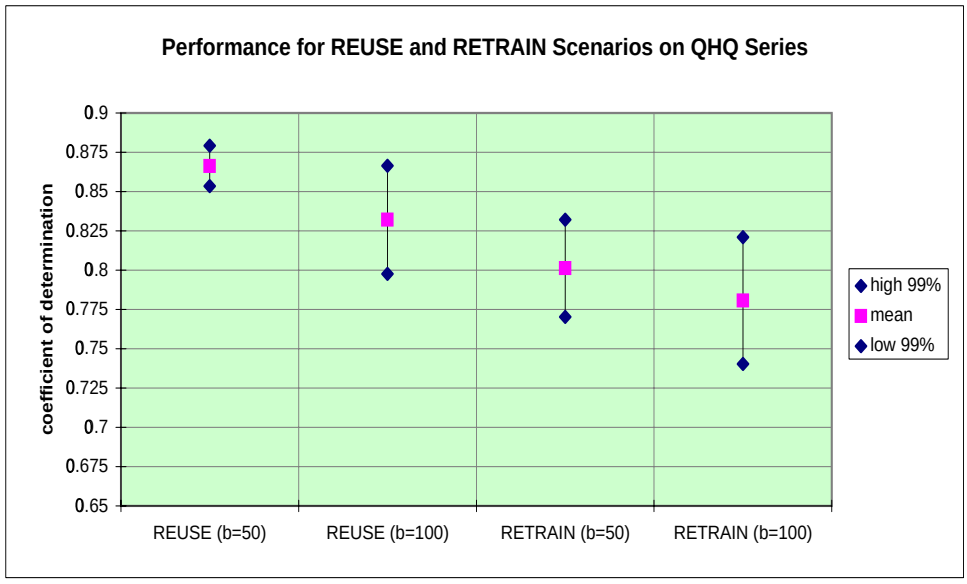

Figure 10.18 Performance for REUSE and RETRAIN Scenarios on QHQ Series

10.3.3.3 The RETRAIN Scenario

The 99% confidence regions for the averaged coefficient of determination obtained in (Drossu and Obradovic, 1996b) using the RETRAIN scenario are shown as the last two bars in Fig. 10.18. Once again, the statistics-based signaling was not considered since the accuracy-based signaling results were better (with small deviations) than those obtained by the statistics-based signaling in the SWITCH scenario. The difference in performance for buffer sizes 50 and 100 was not statistically significant, while the experiments on longer buffer needed less computational resources (3 to 8 retrainings for buffer length 100, as compared to 6 to 16 retrainings for buffer length 50).

In (Drossu and Obradovic, 1996b), additional experimentation was performed on an HQH series (in which the mixing order of the Q and H processes was reversed). To get insight into the robustness of our proposed methodology with respect to the data noise level, two high-noise time series were constructed by corrupting the QHQ and the HQH time series with Gaussian additive noise of zero mean and standard deviation equal to half of the standard deviation of the uncorrupted data. In spite of an extremely high noise level, the accuracy-based signaling technique lead once again to performance that was close to optimal. However, the statistics-based signaling technique was not only significantly less accurate, but not even consistently better than the single-model predictor that used a library model trained entirely on one distribution. As expected, due to a much larger amount of noise, the "optimal" buffer sizes for the accuracy-based signaling were larger as compared to the corresponding ones from the low-noise experiments. The number of neural network retrainings in the high-noise experiments was consistently larger as compared to the low-noise ones, but still reasonably small compared to the length of the time series.

10.4 Conclusions

Neural networks are powerful computational models which have been used in a multitude of time series prediction problems ranging from power consumption (Mangeas et al., 1995) and compressed video traffic (Drossu et al., 1995) to currency exchange (Abu-Mostafa, 1995a). However, because of their inherent complexity, the design of an appropriate neural network predictor for a real-life problem is often a time consuming trial-and-error procedure. In order to shorten the design process, as well as to improve the overall prediction accuracy, different sources of prior knowledge are *directly* embeddable into the neural network models, including, among others, information theory, dynamical system analysis, and stochastic analysis. Each of them has specific advantages but is also prone to variate shortcomings. Nevertheless, the combination of different sources of prior knowledge is likely to provide a more robust predictor, likely to exploit the strengths of each individual knowledge source and to circumvent its weaknesses (Fletcher and Obradovic, 1993).

This was illustrated by embedding stochastic analysis in the neural network design process in the context of an artificially generated, nonlinear, deterministic time series (Mackey-Glass data) and a real-life, non-stationary, stochastic time series (entertainment video traffic data).

In addition to the direct sources of prior knowledge, different *indirectly* embeddable sources are also worth considering. The article briefly illustrated the usefulness of known properties of the target function to be learned and described in more detail different non-stationarity detection methods which incorporate various amounts of prior knowledge and which can significantly improve the efficiency of the neural network predictors. A novel accuracy-based distribution change detection method has been shown to provide significantly more accurate results than traditional statistics-based techniques (Drossu and Obradovic, 1996b).

APPENDIX

A. Minimization of a mutual information upper bound

Applying equation (10.5) to the components of the output vector leads to

$$\begin{aligned} I(y_1; y_2; \ldots ; y_d) &= H(y_1) - H(y_1|y_2, y_3, \ldots y_d) \\ &= H(y_2) - H(y_2|y_1, y_3, \ldots y_d) \\ &\vdots \\ &= H(y_d) - H(y_d|y_1, y_2, \ldots y_{d-1}). \end{aligned} \tag{10.40}$$

Adding up the left and the right hand sides of the previous d expressions for the mutual information, we obtain

$$\begin{aligned} &M = d \cdot I(y_1; y_2; \ldots ; y_d) = \\ &= \textstyle\sum_{i=1}^{d} H(y_i) - \sum_{i=1}^{d} H(y_i|y_1, \ldots , y_{i-1}, y_{i+1}, \ldots , y_d). \end{aligned} \tag{10.41}$$

Applying the chain rule for entropies (Deco and Obradovic, 1996), we obtain

$$M = \sum_{i=1}^{d} H(y_i) - H(\bar{y}). \tag{10.42}$$

It can be shown that the mutual information is always greater or equal to zero, with equality holding only when its variables are independent, thus uncorrelated (Deco and Obradovic, 1996). Therefore, we can express the statistical independence of the components of the output vector as

$$\sum_{i=1}^{d} H(y_i) - H(\bar{y}) = 0. \tag{10.43}$$

Due to the imposed condition of no information loss between input and output in the decorrelation transformation, minimizing the mutual information between the components of the output vector can be reduced to minimizing $\sum_{i=1}^{d} H(y_i)$.

According to Gibbs' second theorem (Deco and Obradovic, 1996), the entropy of a distribution is upper bounded the entropy of a normal distribution with the same variance. Therefore, instead of attempting to minimize $\sum_{i=1}^{d} H(y_i)$, we can attempt to minimize the sum of d Gaussian distributions with individual variances equal to the variances of their corresponding non-Gaussian distributions. Since the entropy of a univariate Gaussian distribution is given by

$$H(X) = \frac{1}{2}\ln(2\pi e\sigma^2), \tag{10.44}$$

with σ^2 being the variance of the Gaussian distribution, the minimization process reduces to minimizing the cost function

$$E = \sum_{i=1}^{d} \ln(\sigma_i^2). \tag{10.45}$$

which can be easily implemented using the gradient descent non-linear optimization technique.

B. Cumulant expansion of the output distribution

A more general approach for decorrelating the components of the output vector is provided by the cumulant expansion of the output distribution. The *moment generating function* or *characteristic function* of a univariate distribution is given as the Fourier transform of its probability density function (Gardiner, 1983),

$$\phi(\omega) = \int p(x)e^{j\omega x}dx, \tag{10.46}$$

where $j = \sqrt{-1}$. We can observe that the derivatives of the moment generating function evaluated at the origin can be expressed in terms of moments (this explains also the function's name), since

$$\phi^{(n)}(\omega) = \frac{\partial^n \phi}{\partial \omega^n} = \int (jx)^n p(x)e^{j\omega x}dx, \tag{10.47}$$

and hence,

$$\phi^{(n)}(0) = j^n m^{(n)}, \tag{10.48}$$

where $m^{(n)}$ represents the moment of order n defined as

$$m^{(n)} = \int x^n p(x)dx. \tag{10.49}$$

Therefore, assuming that all the moments exist and are finite, the moment generating function can be expanded in a power series around the origin, expressed as

$$\phi(\omega) = \sum_{n=0}^{\infty} \frac{j^n \omega^n}{n!} m^{(n)}. \tag{10.50}$$

Similarly, the moment generating function of a multivariate distribution can be expressed as

$$\phi(\bar{\omega}) = \int p(\bar{x}) e^{j\bar{\omega}\cdot\bar{x}} d\bar{x}. \tag{10.51}$$

The inverse Fourier transform of the moment generating function will therefore provide the probability density function,

$$p(\bar{x}) = (2\pi)^{-d} \int \phi(\bar{\omega}) e^{-j\bar{\omega}\cdot\bar{x}} d\bar{\omega}, \tag{10.52}$$

with d representing the dimension of the vector of random variables. In accordance with this inversion formula, the moment generating function determines the probability density function with probability 1. Therefore, the independence condition expressed as

$$p(x_1, x_2, \ldots, x_d) = p_1(x_1) p_2(x_2) \ldots p_d(x_d), \tag{10.53}$$

can also be expressed as

$$\phi(\omega_1, \omega_2, \ldots, \omega_d) = \phi_1(\omega_1)\phi_2(\omega_2)\ldots\phi_d(\omega_d). \tag{10.54}$$

The natural logarithm of the characteristic function is named *cumulant generating function*,

$$\psi(\bar{\omega}) = \ln \phi(\bar{\omega}). \tag{10.55}$$

Assuming that all the higher order moments exist and are finite, we can expand the cumulant generating function in a power series around the origin, expressed as (Gardiner, 1983)

$$\begin{array}{l} \psi(\bar{\omega}) = \sum_{n=1}^{\infty} \frac{j^n}{n!} \sum_{(i)} \ll x_1^{i_1} x_2^{i_2} \ldots x_d^{i_d} \gg \cdot \\ \cdot \omega_1^{i_1} \omega_2^{i_2} \ldots \omega_d^{i_d} \delta\left(n, \sum_{k=1}^{d} i_k\right), \end{array} \tag{10.56}$$

where the quantities $\ll x_1^{i_1} x_2^{i_2} \ldots x_d^{i_d} \gg$ represent the multi-dimensional cumulants of the variables $x_1, x_2, \ldots x_d$, and $\delta\left(n, \sum_{k=1}^{d} i_k\right)$ represents Kronecker's delta function, which is 1 if $\sum_{k=1}^{d} i_k$ equals n and 0 otherwise. In a similar fashion, we can also expand a univariate cumulant generating function in a power series in terms of uni-dimensional cumulants,

$$\psi_i(\omega_i) = \sum_{n=1}^{\infty} \frac{j^n}{n!} \ll x_i^n \gg \omega_i^n, \tag{10.57}$$

with $\ll x_i^n \gg$ representing the uni-dimensional cumulants. The condition 10.54 can

also be expressed in terms of cumulant generating functions as,

$$\ln\left(\phi(\bar{\omega})\right) = \sum_{i=1}^{d} \ln \phi_i(\omega_i), \tag{10.58}$$

which is equivalent to

$$\psi(\bar{\omega}) = \sum_{i=1}^{d} \psi_i(\omega_i). \tag{10.59}$$

The last condition can be made explicit by using (10.56) and (10.57) and hence rewritten as,

$$\begin{array}{l} \sum_{n=1}^{\infty} \frac{j^n}{n!} \sum_{(i)} \ll x_1^{i_1} \dots x_d^{i_d} \gg \omega_1^{i_1} \dots \omega_d^{i_d} \delta\left(n, \sum_{k=1}^{d} i_k\right) = \\ = \sum_{i=1}^{d} \sum_{n=1}^{\infty} \frac{j^n}{n!} \ll x_i^n \gg \omega_i^n \end{array} \tag{10.60}$$

The method of computing the multi-dimensional cumulants $\ll x_1 x_2 \dots x_n \gg$ of any desired order n can be presented in an algorithmic fashion as (Gardiner, 1983):

- Write a sequence of n dots, ...
- Divide this sequence into $p+1$ subsequences, each enclosed in angular brackets, with p varying from 0 to $n-1$,

$$< .. >< ... >< ..... > \dots < . > . \tag{10.61}$$

- Replace the dots by the symbols $x_1, \dots, x_n$, in such a fashion that all the *different* expressions occur, thus

$$\begin{array}{l} < x_1 >< x_2 x_3 >= \ < x_1 >< x_3 x_2 > \neq \\ \neq \ < x_3 >< x_1 x_2 >, \end{array} \tag{10.62}$$

where

$$< x_1 x_2 \dots x_n >= \int x_1 x_2 \dots x_n p\left(\bar{x}\right) d\bar{x} \tag{10.63}$$

$$< x_i^n >= \int x_i^n p(x_i) dx_i, \tag{10.64}$$

with $\bar{x} = (x_1, x_2, \dots, x_n)$.

- For every p, take the sum of all the terms containing $p+1$ subsequences and call this sum $C_p(x_1, x_2, \dots, x_n)$.
- Compute the multi-dimensional cumulant as

$$\ll x_1 x_2 \dots x_n \gg \ = \ \sum_{p=0}^{n-1} (-1)^p \, p! C_p(x_1, x_2, \dots, x_n). \tag{10.65}$$

- In cases in which a cumulant contains a repeated term, like $\ll x_1^2 x_2 x_3 \gg$, compute $\ll x_1 x_2 x_3 x_4 \gg$ and in the resulting expression set $x_4 = x_1$.

Although cumulants of whichever desired order can be theoretically computed, in practice cumulants of order four will rarely be exceeded. The first four multi-dimensional cumulants can be expressed as,

$$
\begin{aligned}
&\ll x_i \gg \;=\; < x_i > \\
&\ll x_i x_j \gg \;=\; < x_i x_j > - < x_i >< x_j > \\
&\ll x_i x_j x_k \gg \;=< x_i x_j x_k > - < x_i x_j >< x_k > - \\
&- < x_j x_k >< x_i > - < x_k x_i >< x_j > + \\
&+2 < x_i >< x_j >< x_k > \\
&\ll x_i x_j x_k x_l \gg \;=< x_i x_j x_k x_l > - < x_i x_j x_k >< x_l > - \\
&- < x_j x_k x_l >< x_i > - < x_k x_l x_i >< x_j > - \\
&- < x_l x_i x_j >< x_k > - < x_i x_j >< x_k x_l > - \\
&- < x_i x_k >< x_j x_l > - < x_i x_l >< x_j x_k > + \\
&+2 < x_i x_j >< x_k >< x_l > +2 < x_i x_k >< x_j >< x_l > + \\
&+2 < x_i x_l >< x_j >< x_k > +2 < x_j x_k >< x_i >< x_l > + \\
&+2 < x_j x_l >< x_i >< x_k > +2 < x_k x_l >< x_i >< x_j > - \\
&-6 < x_i >< x_j >< x_k >< x_l >,
\end{aligned}
\tag{10.66}
$$

whereas the uni-dimensional cumulants up to fourth order are

$$
\begin{aligned}
&\ll x_i^1 \gg \;=\; < x_i > \\
&\ll x_i^2 \gg \;=\; < x_i^2 > - < x_i >^2 \\
&\ll x_i^3 \gg \;=\; < x_i^3 > -3 < x_i^2 >< x_i > +2 < x_i >^3 \\
&\ll x_i^4 \gg \;=< x_i^4 > -4 < x_i^3 >< x_i > - \\
&-3 < x_i^2 >^2 +12 < x_i^2 >< x_i >^2 -6 < x_i >^4 .
\end{aligned}
\tag{10.67}
$$

Replacing the x_i's by y_i's, to indicate the components of the output vector, the independence condition (10.60) can be rewritten by replacing the expressions for the multi-dimensional and the uni-dimensional cumulants, resulting in

$$
\begin{aligned}
&-\tfrac{1}{2} \textstyle\sum_{i,j} \omega_i \omega_j \left\{ \ll y_i y_j \gg - \ll y_i^2 \gg \delta_{ij} \right\} - \\
&-\tfrac{j}{6} \textstyle\sum_{i,j,k} \omega_i \omega_j \omega_k \left\{ \ll y_i y_j y_k \gg - \ll y_i^3 \gg \delta_{ijk} \right\} + \\
&+\tfrac{1}{24} \textstyle\sum_{i,j,k,l} \omega_i \omega_j \omega_k \omega_l \left\{ \ll y_i y_j y_k y_l \gg - \ll y_i^4 \gg \delta_{ijkl} \right\} = \\
&= 0.
\end{aligned}
\tag{10.68}
$$

Considering additionally that the mean of the output vector has been removed, the previous condition can be further expressed as (Deco and Schurmann, 1995)

$$
\begin{aligned}
&-\tfrac{1}{2} \textstyle\sum_{i,j} \omega_i \omega_j \left\{ < y_i y_j > - < y_i^2 > \delta_{ij} \right\} \\
&-\tfrac{j}{6} \textstyle\sum_{i,j,k} \omega_i \omega_j \omega_k \left\{ < y_i y_j y_k > - < y_i^3 > \delta_{ijk} \right\} \\
&+\tfrac{1}{24} \textstyle\sum_{i,j,k,l} \omega_i \omega_j \omega_k \omega_l \left\{ < y_i y_j y_k y_l > \right. \\
&\left. -3 < y_i y_j >< y_k y_l > - \left(< y_i^4 > -3 < y_i^2 >^2 \right) \delta_{ijkl} \right\} = 0.
\end{aligned}
\tag{10.69}
$$

The $\delta_{i\ldots j}$ denote Kronecker's delta, which equals 1 only when all the subscripts are equal to each other and equals 0 otherwise. Since the previous relation has to be satisfied for all $\bar{\omega}$, the terms inside each summation must be equal to zero. Hence, for all i, j, k, l

$$\begin{aligned} &< y_i y_j > - < y_i^2 > \delta_{ij} = 0, \\ &< y_i y_j y_k > - < y_i^3 > \delta_{ijk} = 0, \\ &< y_i y_j y_k y_l > -3 < y_i y_j >< y_k y_l > \\ &- \left(< y_i^4 > -3 < y_i^2 >^2 \right) \delta_{ijkl} = 0. \end{aligned} \tag{10.70}$$

According to (Deco and Schurmann, 1995), the previous conditions can be expressed in the equivalent form

$$\begin{aligned} &< y_i y_j >= 0, \quad \text{if } (i \neq j), \\ &< y_i y_j y_k >= 0, \quad \text{if } (i \neq j \vee i \neq k), \\ &< y_i y_j y_k y_l >= 0, \quad \text{if } (\{i \neq j \vee i \neq k \vee i \neq l\} \wedge \neg L), \\ &< y_i^2 y_j^2 > -3 < y_i^2 >< y_j^2 >= 0, \quad \text{if } (i \neq j), \end{aligned} \tag{10.71}$$

with L being the logical expression

$$\begin{aligned} L = \{ \quad &(i = j \wedge k = l \wedge j \neq k) \vee \\ &(i = k \wedge j = l \wedge i \neq j) \vee \\ &(i = l \wedge j = k \wedge i \neq j) \ \}. \end{aligned} \tag{10.72}$$

These conditions can be imposed by using the gradient descent non-linear optimization applied to the cost function

$$\begin{aligned} &E = \alpha \textstyle\sum_{i<j} < y_i y_j >^2 + \beta \sum_{i<j\leq k} < y_i y_j y_k >^2 + \\ &+ \gamma \textstyle\sum_{i<j\leq k\leq l} < y_i y_j y_k y_l >^2 + \\ &+ \delta \textstyle\sum_{i<j} \left(< y_i^2 y_j^2 > -3 < y_i^2 >< y_j^2 > \right)^2, \end{aligned} \tag{10.73}$$

with α, β, γ, and δ, representing the inverses of the number of elements in their corresponding summations.

C. AR(p) Approximation by a Neural Network

Consider the approximation of an AR model of order p by a feedforward neural network with p input units, p hidden units and a single output unit. It is assumed that each hidden unit uses an activation function of the form $f(x) = 1/(1 + e^{-\beta x})$, whereas the output unit uses the identity function $i(x) = x$ as its activation function. In this neural network, for all $i, j \in \{1, \ldots, p\}$, let us set the hidden unit biases to

$$\theta_i = 0, \tag{10.74}$$

and the input-to-hidden layer weights to

$$w_{ij} = \delta_{ij}, \tag{10.75}$$

where δ_{ij} is Kronecker's delta function. Using the notation from Fig. 10.2, on input $(x_{t-1}, \ldots, x_{t-p})$, the neural network output can be written as

$$\hat{x}_t = \sum_{i=1}^{p} \left(\frac{W_i}{1+e^{-\beta x_{t-i}}} + \gamma_i \right) = \sum_{i=1}^{p} g(x_{t-i}), \tag{10.76}$$

where $\sum_{i=1}^{p} \gamma_i = \Gamma$. On the same input, the AR(p)-based predictor outputs

$$\hat{x}_t = \sum_{i=1}^{p} \varphi_i x_{t-i} = \sum_{i=1}^{p} h(x_{t-i}). \tag{10.77}$$

Expression (10.76) approximates (10.77) for any combination of inputs that are small enough if each $g(x_{t-i})$ approximates the corresponding $h(x_{t-i})$. Expanding $g(x_{t-i})$ in a Taylor series around the origin and keeping just the terms up to order 1, we obtain

$$g(x_{t-i}) \simeq g(0) + g'(0)x_{t-i} = \frac{W_i}{2} + \gamma_i + \frac{\beta W_i}{4} x_{t-i} \tag{10.78}$$

Hence, setting $g(x_{t-i}) = h(x_{t-i})$ leads to

$$W_i = \frac{4\varphi_i}{\beta} \tag{10.79}$$

$$\gamma_i = -\frac{W_i}{2}, \tag{10.80}$$

So, the neural network with p inputs, p hidden units and interconnection parameters

$$\begin{aligned} &w_{ij} = \delta_{ij} \\ &\theta_i = 0 \\ &W_i = \tfrac{4}{\beta}\varphi_i \\ &\Gamma = -\tfrac{2}{\beta}\textstyle\sum_{i=1}^{p} \varphi_i, \end{aligned} \tag{10.81}$$

where $i, j \in \{1, \ldots, p\}$, approximates an AR model of order p with parameters $\varphi_1, \ldots, \varphi_p$.

For the approximation (10.78) to be reasonably accurate, x_{t-i} has to be close to zero. For $x_{t-i} \in [-1, 1]$, the maximum relative error when approximating $h(x_{t-i})$ (given in (10.77)) by $g(x_{t-i})$ (given in (10.76)), with W_i and γ_i computed according to (10.79) and (10.80), respectively, is 8% for $\beta = 1$, 2% for $\beta = 0.5$ and 0.08% for $\beta = 0.1$.

D. Chi-square and Kolmogorov-Smirnov Tests

In the *chi-square test*, the data range of the two data sets to be compared is divided into a number of intervals (bins). Assuming that R_i and S_i represent the number of data samples in bin i for the first and the second data set, respectively, the chi-square statistic computes

$$\chi^2 = \sum_i \frac{(R_i - S_i)^2}{R_i + S_i},$$

with the sum taken over all bins. The complement of the incomplete gamma function,

$$Q(\nu, \chi^2) = \frac{1}{\Gamma(\nu)} \int_{\chi^2}^{\infty} e^{-t} t^{a-1} dt,$$

where

$$\Gamma(x) = \int_0^{\infty} t^{x-1} e^{-t} dt,$$

is then evaluated and a small value of Q (close to 0) indicates that it is unlikely that the two distributions are the same. Here, ν represents the number of degrees of freedom which in the case when the two sets have the same number of data samples ($\sum R_i = \sum S_i$), equals the number of bins minus one. If the previous restriction is not imposed, than ν equals the number of bins.

The *Kolmogorov-Smirnov (K-S) test* measures the absolute difference between two cumulative distribution functions S_{N_1} and S_{N_2} with N_1 and N_2 data points, respectively. The K-S statistic computes

$$D = \max_{-\infty < x < \infty} |S_{N_1}(x) - S_{N_2}(x)|.$$

The function Q_{KS} defined as

$$Q_{KS}(\lambda) = 2 \sum_{j=1}^{\infty} (-1)^{j-1} e^{-2j^2\lambda^2}$$

is computed for

$$\lambda = D(\sqrt{N_e} + 0.12 + 0.11/\sqrt{N_e}),$$

where N_e is the effective number of data points,

$$N_e = \frac{N_1 N_2}{N_1 + N_2}.$$

A small value of Q_{KS} (close to 0) indicates that it is unlikely that the two distributions are the same.

References

Abecasis, S. M. and Lapenta, E. S., 1996. Nonstationary Time-Series Forecasting within a Neural Network Framework. *NeuroVe$t Journal* 4, no. 4:9–16.

Abu-Mostafa, Y., 1995a. Financial Applications of Learning from Hints. In *Advances in Neural Information Processing Systems*, ed. G. T. et al., vol. 7, pp. 411–418.

Abu-Mostafa, Y., 1995b. Hints. *Neural Computation* 7:639–671.

Anderson-Sprecher, R., 1994. Model Comparisons and R^2. *The American Statistician* 48, no. 2:113–117.

Barlow, H. B., 1989. Unsupervised Learning. *Neural Computation* 1:295–311.

Barron, A. R., 1993. Universal Approximation Bounds for Superposition of a Sigmoidal Function. *IEEE Transactions on Information Theory* 39, no. 3:930–945.

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C., 1994. *Time Series Analysis. Forecasting and Control. Third Edition*. Prentice Hall.

Chenoweth, T. and Obradovic, Z., 1996. A Multi-Component Nonlinear Prediction System for the S&P 500 Index. *Neurocomputing* 10, no. 3:275–290.

Connor, J. T., Martin, R. D., and Atlas, L. E., 1994. Recurrent Neural Networks and Robust Time Series Prediction. *IEEE Transactions on Neural Networks* 5, no. 2:240–254.

Cybenko, G., 1989. Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signal, and Systems* 2:303–314.

Deco, G. and Brauer, W., 1995. Nonlinear Higher-Order Statistical Decorrelation by Volume-Conserving Neural Architectures. *Neural Networks* 8, no. 4:525–535.

Deco, G. and Obradovic, D., 1996. *An Information-Theoretic Approach to Neural Computing*. Springer.

Deco, G. and Schurmann, B., 1995. Learning Time Series Evolution by Unsupervised Extraction of Correlations. *Physical Review E* 51, no. 3:1780–1790.

Drossu, R., Lakshman, T. V., Obradovic, Z., and Raghavendra, C., 1995. Single and Multiple Frame Video Traffic Prediction Using Neural Network Models. In *Computer Networks, Architecture and Applications*, eds. S. V. Raghavan and B. N. Jain, pp. 146–158. Chapman and Hall.

Drossu, R. and Obradovic, Z., 1996a. Efficient Design of Neural Networks for Time Series Prediction. *IEEE Computational Science and Engineering* 3, no. 2:78–89.

Drossu, R. and Obradovic, Z., 1996b. Regime Signaling Techniques for Non-Stationary Time-Series Forecasting. *NeuroVe$t Journal* 4, no. 5:7–15.

Fahlman, S. and Lebiere, C., 1990. The Cascade-Correlation Learning Architecture. In *Advances in Neural Information Processing Systems*, ed. D. S. Touretzky, vol. 2, pp. 524–532.

Fletcher, J. and Obradovic, Z., 1993. Combining Prior Symbolic Knowledge and Constructive Neural Networks. *Connection Science: Journal of Neural Computing, Artificial Intelligence and Cognitive Research* 5, no. 3-4:365–375.

Gardiner, C. W., 1983. *Handbook of Stochastic Methods*. Springer.

Grassberger, P. and Procaccia, I., 1983. Measuring the Strangeness of Strange Attractors. *Physica D* 9:189–208.

Haykin, S., 1994. *Neural Networks. A Comprehensive Foundation*. MacMillan.

Jurgens, H. O. P. H. and Saupe, D., 1992. *Chaos and Fractals. New Frontiers of Science*. Springer.

Lapedes, A. and Farber, R., 1987. Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling. *Technical Report, LA-UR87-2662, Los Alamos National Laboratory* .

Liebert, W. and Schuster, H. G., 1989. Proper Choice of Time Delay for the Analysis of Chaotic Time Series. *Physics Letters A* 142:107–111.

Mangeas, M., Muller, C., and Weigend, A. S., 1995. Forecasting Electricity Demand Using Nonlinear Mixture of Experts. In *World Congress on Neural Networks*, vol. 2, pp. 48–53.

Papoulis, A., 1984. *Probability, Random Variables, and Stochastic Processes. Second Edition*. McGraw-Hill.

Pineda, F. and Sommerer, J. C., 1993. Estimating Generalized Dimensions and Choosing Time Delays: A Fast Algorithm. In *Time Series Prediction: Forecasting the Future and Understanding the Past*, eds. A. S. Weigend and N. A. Gershenfeld, pp. 367–385. Addison-Wesley.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., 1992. *Numerical Recipes in C. Second Edition*. Cambridge University Press.

Redlich, A. N., 1993a. Redundancy Reduction as a Strategy for Unsupervised Learning. *Neural Computation* 5:289–304.

Redlich, A. N., 1993b. Supervised Factorial Learning. *Neural Computation* 5:750–766.

Sjoberg, J., Hjalmarsson, H., and Ljung, L., 1994. Neural Networks in System Identification. In *Proc. 10th IFAC Symposium on System Identification (SYSID) '94*, vol. 2, pp. 49–72. Copenhagen, Denmark.

Sjoberg, J., Zhang, Q., Ljung, L., Benveniste, A., and et al., B. D., 1995. Nonlinear

Black-Box Modeling in System Identification: a Unified Overview. *Automatica* 31, no. 12:1691–1724.

Takens, F., 1981. Detecting Strange Attractors in Turbulence. In *Lecture Notes in Mathematics*, eds. D. Rand and L. Young, pp. 366–381. Springer.

Weigend, A. S., Mangeas, M., and Srivastava, A., 1995. Nonlinear Gated Experts for Time Series: Discovering Regimes and Avoiding Overfitting. *International Journal of Neural Systems* 6:373–399.

Werbos, P., 1992. Neural Networks, System Identification and Control in the Chemical Process Industries. In *Handbook of Intelligent Control. Neural, Fuzzy, and Adaptive Approaches*, eds. D. A. White and D. A. Sofge, pp. 283–356. Van Nostrand Reinhold.

Werbos, P., 1994. *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. John Wiley and Sons.

Wolf, A., Swift, J. B., Swinney, H. L., and Vastano, J. A., 1985. Determining Lyapunov Exponents from a Time Series. *Physica D* 16:285–317.

Radu Drossu received his M.S. degree in Electrical Engineering from the Polytechnical Institute of Bucharest - Romania in 1990 and his Ph.D. degree in Computer Science from Washington State University in 1997, doing his research on the efficient design of neural networks for time series prediction problems under the supervision of Dr. Zoran Obradović.

Zoran Obradović received his B.S. degree in Applied Mathematics, Information and Computer Sciences in 1985, his M.S. degree in Mathematics and Computer Science in 1987, both from the University of Belgrade and his Ph.D. degree in Computer Science from the Pennsylvania State University in 1991. He is currently an associate professor in the School of Electrical Engineering and Computer Science, Washington State University, and an adjunct research scientist at the Mathematical Institute of the Serbian Academy of Sciences and Arts. The objective of his research is to explore the applicability of neural network technology to large scale classification and time series prediction problems in very noisy domains.