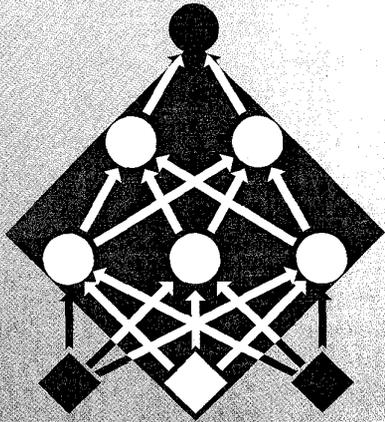


## NEURAL NETWORKS



Can a neural network be rapidly designed to predict a time series? Stochastic analysis can provide some initial knowledge about an appropriate NN architecture, parameter values, and data sampling rate. Such rapidly devised neural networks are not optimal, but still perform similarly to NNs designed more elaborately through an expensive trial-and-error procedure.

# Rapid Design of Neural Networks for Time Series Prediction

Radu Drossu and Zoran Obradovic  
*Washington State University*

It is human nature to want to know in advance what is likely to happen in the future. Observing past outcomes of a phenomenon in order to anticipate its future behavior represents the essence of *forecasting* or *prediction*. If a complete mathematical model describing a studied phenomenon is known and not overly complex, and if the initial conditions are sufficiently defined, forecasting often becomes a trivial task. However, when an analytical model is unknown, incomplete, or too complex, a typical alternative is to try to forecast by building a model that takes into account only previous outcomes of the phenomenon while ignoring any exterior influence. In other words, look at *what* it does, not *why*. For example, the simplest attempt to predict US electric power consumption would be to build a prediction model based just on previous values of power consumption, neglecting any other information that might be available (temperature, time of the day, season, and so on). The outcomes of the phenomenon over time form a *time series*.

More formally, a time series  $\{x_t\}$  can be defined as a function  $x$  of an independent variable  $t$ , stemming from a process for which a mathematical description is unknown. The main characteristic of such a series is that its future behavior cannot be predicted exactly, as can that of a known deterministic function of  $t$ . However, the behavior of a time series can sometimes be anticipated by describing it through probabilistic laws. Commonly, time series prediction problems are approached either from a stochastic perspective<sup>1</sup> or, more recently, from a neural network perspective.<sup>2</sup> Each of these approaches has advantages and disadvantages: stochastic methods are usually fast, but of limited applicability since they commonly employ only linear models. The NN methods, on the other hand, are powerful enough, but selecting an appropriate architecture and parameters is a time-consuming trial-and-error procedure.

## Why use NNs for time series prediction?

Though the relationship may not be immediately apparent, NNs might be very well suited to modeling time series for at least two reasons:

- (1) Theoretical work shows that NNs are powerful enough to uniformly approximate almost any arbitrary continuous function on a compact domain,<sup>3</sup> similar to traditional universal approximation techniques based on Taylor function expansion, Fourier series, and so forth. However, in addition to their ability to represent complex nonlinear functions, NNs can effectively construct approximations for unknown functions by learning from examples (known outcomes of the function). This ability to approximate unknown complex input-output mappings makes them attractive in practical applications where traditional computational structures have performed poorly (such as those with ambiguous data or large contextual influence). NN models can be complemented by other successful approximation techniques based on wavelets, kernel estimators, nearest neighbors, B-splines, hinging hyperplanes, projection pursuit regression, partial least squares, and fuzzy models.<sup>4</sup>
- (2) There is a direct relationship between the basic stochastic models for time series and NN models, as we will explain.

Other researchers<sup>5</sup> have compared stochastic and NN techniques for time series prediction with respect to the time series history required for building a reliable model and the decrease in prediction accuracy when predicting further into the future. The combination of stochastic and NN techniques in a hybrid system for improving prediction accuracy, or the use of some stochastic prior knowledge of the underlying time series for configuring the NN, are topics that deserve further consideration.

This article explores the possibility of rapidly designing an appropriate NN for time series prediction based on information obtained from stochastic modeling. Such an analysis could provide some initial knowledge regarding the choice of an NN architecture and parameters, as well as regarding an appropriate data-sampling rate. Stochastic analysis provides a complementary approach to previously proposed dynamical system analysis for NN design. Based on Takens's theorem,<sup>6</sup> an estimate of the di-

mension  $m$  of the manifold from which the time series originated can be used to construct an NN model using  $2m + 1$  external inputs.<sup>7</sup> This design is further extended by Potts and Broomhead,<sup>8</sup> who first embed the state space of a discrete-time dynamical system in a manifold of dimension  $n \gg 2m + 1$ , which is further projected to its  $2m + 1$  principal components used as external inputs in a radial-basis-function NN model for time series prediction.

Our approach is to perform an initial stochastic analysis of the data and to choose an appropriate NN architecture, and possibly initial values for the NN parameters, according to the most adequate linear model. This idea is supported by Juditsky et al., who remark that "many nonlinear 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 NN model."<sup>9</sup> The motivation for this approach is that it is much more cost-effective to select an NN architecture with the help of linear stochastic modeling than by trial and error. The objective of this study is not to obtain "the optimal" NN architecture for a given problem, but to rapidly provide an architecture with close-to-optimal performance. Since information is obtained from a linear model, for more complex problems the NN might be overdimensioned (similar performance could be obtained using a smaller machine and fewer learning examples). However, the exhaustive trial-and-error procedure involved for determining such an optimal machine could be costlier than the alternative based on stochastic analysis.

We evaluate this approach in the context of different prediction objectives, as well as data sets of varying complexity. An additional issue we address is whether NNs different from the ones suggested by the stochastic model can lead to significant improvements in prediction accuracy, as compared to the suggested ones.

## Stochastic models: ARMA and NARMA

Stationary time series can be described as time series whose characteristic parameters are invariant in time. This makes them very attractive in practice, since it implies that they could be represented by time-invariant models. The first step in stochastic modeling is thus an attempt to "stationarize" the studied time series through a suitable discrete differentiation preprocessing, as described later.

A general linear stochastic model of a stationary time series is the *autoregressive moving average* model of orders  $p$  and  $q$ , denoted as ARMA( $p, q$ ). It describes the process value as a weighted sum of (1)  $p$  previous values of the given process and (2) the current as well as  $q$  previous values of a separate, and random, process. Formally, a stationary ARMA( $p, q$ ) process  $\{x_t\}$  with zero mean is represented as

$$x_t = \varphi_1 x_{t-1} + \varphi_2 x_{t-2} + \dots + \varphi_p x_{t-p} + a_t + \psi_1 a_{t-1} + \psi_2 a_{t-2} + \dots + \psi_q a_{t-q} \quad (1)$$

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

$$\hat{x}_t = \varphi_1 x_{t-1} + \varphi_2 x_{t-2} + \dots + \varphi_p x_{t-p} + \psi_1 a_{t-1} + \psi_2 a_{t-2} + \dots + \psi_q a_{t-q} \quad (2)$$

The error between the real process value  $x_t$  and the predicted value  $\hat{x}_t$  is called the *residual*. If the ARMA( $p, q$ ) predictor is adequate, the residual should have the same statistics as the random

process,  $a_t$ , appearing in the description (Equation 1) of its underlying ARMA( $p, q$ ) model.

Some time series might be modeled fairly accurately by using special, simpler cases of the ARMA( $p, q$ ) model: the AR( $p$ ) or MA( $q$ ) models. The AR( $p$ ) model is described as

$$x_t = \varphi_1 x_{t-1} + \varphi_2 x_{t-2} + \dots + \varphi_p x_{t-p} + a_t \quad (3)$$

and the MA( $q$ ) is described as

$$x_t = a_t + \psi_1 a_{t-1} + \psi_2 a_{t-2} + \dots + \psi_q a_{t-q} \quad (4)$$

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

$$x_t = b(x_{t-1}, x_{t-2}, \dots, x_{t-p}, a_{t-1}, \dots, a_{t-q}) + a_t \quad (5)$$

and the NAR model

$$x_t = b(x_{t-1}, x_{t-2}, \dots, x_{t-p}) + a_t \quad (6)$$

where  $b$  is an unknown smooth function.

The AR-, MA-, NARMA-, and NAR-based predictors are obtained from their corresponding models in a way analogous to obtaining the ARMA-based predictor (Equation 2) from the ARMA model (Equation 1).

The NARMA and NAR models are very complex, thus being unsuitable for real-life applications. Fortunately, they are closely related to more practical nonlinear models: the *neural networks*.

### Approximating stochastic models with neural networks

Neural networks consist of a number of relatively simple processing units called *neurons*. The neurons are interconnected through synaptic links, called *weights*, and are grouped in *layers*, with synaptic links connecting neurons in adjacent layers. Three different layer types can be distinguished: *input layer* (the layer that external stimuli are applied to), *output layer* (the layer that outputs results to the exterior world), and one or more *hidden layers* (in-

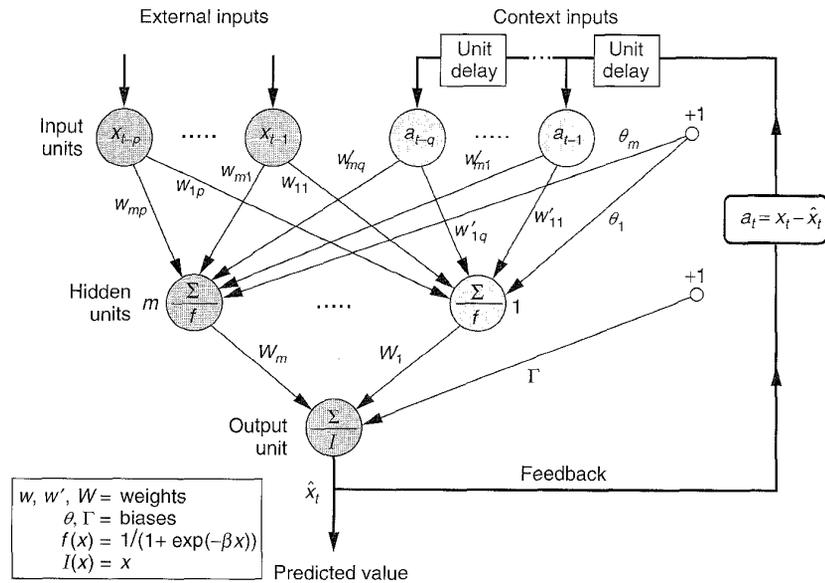


Figure 1. Approximating a nonlinear stochastic predictor of a time series (NARMA predictor) with a three-layer recurrent neural network.

intermediate computational layers between input and output). The NN architectures considered here are either *feedforward*, in which the signal flow is from input layer towards output layer, or *recurrent*, in which the feedforward signal flow is supplemented with additional feedback connections from output to input layer. Each neuron in the hidden and output layers applies an *activation function* (usually a nonlinear smooth and bounded function) to the sum of its weighted inputs and a neuron-specific parameter (called *bias*). [See *CS&E* Spring '96, p. 36 and *Computer* Mar. '96, pp. 24, 31 for further reading on the basics of NNs. —Ed.]

Recurrent and feedforward NNs have been proposed<sup>2,10</sup> for simulating NARMA and NAR models respectively. An invertible<sup>1</sup> NARMA-based predictor can be approximated as

$$\begin{aligned} \hat{x}_t &= b(x_{t-1}, \dots, x_{t-p}, a_{t-1}, \dots, a_{t-q}) \\ &\approx \sum_{i=1}^m W_i f \left( \sum_{j=1}^p w_{ij} x_{t-j} \right. \\ &\quad \left. + \sum_{j=1}^q w'_{ij} (x_{t-j} - \hat{x}_{t-j}) + \theta_i \right) + \Gamma \end{aligned} \quad (7)$$

where  $f$  represents a nonlinear, smooth, and bounded function and  $a_k = x_k - \hat{x}_k$  for all  $k \in \{t-q, \dots, t-1\}$ . This approximation of the NARMA-based model corresponds to the recurrent NN in Figure 1, in which  $W_i$  are the weights between hidden and output neurons,  $w_{ij}$  are the weights between external-input neurons and hidden neurons,  $w'_{ij}$  are the weights between context-input neurons and hidden neurons,  $\theta_i$  are the hidden-neuron biases,  $\Gamma$  is the output-neuron bias, and  $f$  is the activation function of the hidden neurons. Similarly, a NAR-based predictor can be approximated as

$$\begin{aligned} \hat{x}_t &= b(x_{t-1}, \dots, x_{t-p}) \\ &\approx \sum_{i=1}^m W_i f \left( \sum_{j=1}^p w_{ij} x_{t-j} + \theta_i \right) + \Gamma \end{aligned} \quad (8)$$

obtained by disconnecting the context inputs  $a_{t-1} \dots a_{t-q}$  from Figure 1.

The single-hidden-layer NN architecture can be extended in a straightforward manner to one with multiple hidden layers by inserting additional layers of neurons, with their associated synaptic links, between the existing hidden layer and the output layer.

A special case of particular interest in this study is the approximation of a linear AR predictor by a feedforward NN. If we removed the hidden layer and the feedback connections from the NN in Figure 1 it would be computationally equivalent to the linear AR predictor. However, such a trivial NN would be of no interest since it could not do any better than the equivalent AR model. Instead, we consider approximating an AR model of order  $p$  by a feedforward NN with  $p$  input units,  $p$  hidden units, and a single output unit. 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 NN, let us set the weights between input and hidden layer as

$$w_{ij} = \delta_{ij}, \quad \text{for all } i, j \in \{1, \dots, p\} \quad (9)$$

and the hidden neurons' biases as

$$\theta_i = 0, \quad \text{for all } i \in \{1, \dots, p\} \quad (10)$$

where  $j$  represents the number of the input neuron,  $i$  represents the number of the hidden neuron, and  $\delta_{ij} = 1$  if  $i = j$  and  $\delta_{ij} = 0$  otherwise.

Using the notation from Figure 1, on input  $(x_{t-1}, \dots, x_{t-p})$ , the NN output can be written as

$$\hat{x}_t = \sum_{i=1}^p \left( \frac{W_i}{1 + e^{-\beta x_{t-i}}} + \gamma_i \right) = \sum_{i=1}^p g(x_{t-i}) \quad (11)$$

where  $\sum_{i=1}^p \gamma_i = \Gamma$ .

On the same input, the AR( $p$ )-based predictor outputs

$$\hat{x}_t = \sum_{i=1}^p \varphi_i x_{t-i} = \sum_{i=1}^p b(x_{t-i}) \quad (12)$$

Expression 11 approximates 12 if each  $g(x_{t-i})$  approximates the corresponding  $b(x_{t-i})$ . Expanding  $g(x_{t-i})$  in a Taylor series around the origin and keeping just the terms up to order 1, we obtain

$$\begin{aligned} g(x_{t-i}) &\approx g(0) + g'(0)x_{t-i} \\ &= \frac{W_i}{2} + \gamma_i + \frac{\beta W_i}{4} x_{t-i} \end{aligned} \quad (13)$$

Hence, by setting  $g(x_{t-i}) = b(x_{t-i})$ , we obtain

$$\gamma_i = -\frac{W_i}{2} \quad \text{and} \quad (14)$$

$$W_i = \frac{4\phi_i}{\beta} \tag{15}$$

So, the NN with  $p$  inputs,  $p$  hidden units, and interconnection parameters

$$\begin{aligned} w_{ij} &= \delta_{ij} && \text{for all } i, j \in \{1, \dots, p\}, \\ \theta_i &= 0 && \text{for all } i \in \{1, \dots, p\}, \\ W_i &= \frac{4}{\beta} \phi_i && \text{for all } i \in \{1, \dots, p\}, \text{ and} \\ \Gamma &= -\frac{2}{\beta} \sum_{i=1}^p \phi_i \end{aligned} \tag{16}$$

approximates an AR predictor of order  $p$  with parameters  $\phi_1, \dots, \phi_p$ .

For the approximation in Equation 13 to be reasonably accurate,  $x_{t-i}$  has to be close to zero. For  $x_{t-i} \in [-1, 1]$ , the maximum relative errors when approximating  $h(x_{t-i})$  from Equation 12 by  $g(x_{t-i})$  from Equation 11 (with  $\gamma_i$  and  $W_i$  computed according to Equations 14 and 15) are 8, 2, and 0.08 percent for  $\beta = 1, 0.5$ , and  $0.1$ , respectively.

Commonly, the parameters  $w_{ij}$ ,  $w'_{ij}$ ,  $W_i$ ,  $\theta_i$ , and  $\Gamma$  are estimated from examples by a gradient-descent error minimization technique known as backpropagation learning.<sup>11</sup> This is also the learning method employed in our experiments.

### Time series prediction process

Both the NN and the ARMA prediction processes proceed as follows:

- (1) Preprocess the data
- (2) Identify the model to use
- (3) Estimate parameters using a first data set
- (4) Validate the model using a second data set
- (5) Predict the time series

Steps 2 through 4 are iterated until the model is suitable. The individual steps are performed as follows.

#### Step 1

The data preprocessing step usually comprises *smoothing* and possibly *stationarization*. In practice a logarithmic transformation of the original positive-valued series is commonly performed for smoothing ( $y_t = \log(x_t)$ ) and a first- or second-order discrete differentiation for stationarization ( $z_t = y_t - y_{t-1}$  or  $z_t = y_t - 2y_{t-1} + y_{t-2}$  respectively).

#### Step 2

For neural network prediction, the model identification step selects an NN architecture

(feedforward or recurrent), a layer structure (number of layers and number of units per layer), and learning parameters (learning rate, momentum, and tolerance).

For ARMA prediction, this step selects a model type (AR, MA, or ARMA), as well as corresponding model orders.

#### Step 3

For neural networks, the parameter estimation step encompasses the NN training on a first data set (training set), in which the network weights are modified according to a given learning technique (backpropagation in our case).

For ARMA, this step consists of a maximum likelihood estimation of the model parameters on a first data set (in the case of pure AR models, the Modified Covariance method or Burg's method are also applicable<sup>12</sup>).

#### Step 4

The model validation step checks the adequacy of the model by performing the residual analysis of prediction errors on the second data set. Alternatively, in the absence of a second data set, the model validation should encompass the analysis of Akaike's final prediction error (FPE) and/or information criterion (AIC), according to which the most appropriate model is the one that yields the smallest values for FPE or AIC.

Although the prediction error plot is commonly encountered in the residual analysis, it only provides a subjective means of visually evaluating the prediction accuracy. It is much more desirable to have a quantitative means of evaluating the prediction. For this reason, the residual analysis of prediction errors should comprise at least the computation of

- ◆ the *error mean*

$$\mu = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i), \text{ and}$$

- ◆ the *coefficient of determination*

$$r^2 = 1 - \frac{\sum_{i=1}^n (x_i - \hat{x}_i)^2}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

where  $x_i$  and  $\hat{x}_i$  denote actual and predicted process values, respectively, while  $\bar{x}$  denotes the mean of the actual data.

The error mean  $\mu$  is usually used just to measure whether the predictor is biased or not. A predictor with error mean close to zero is called

unbiased, whereas a predictor with error mean far from zero indicates a biased predictor, and points out that there are some deficiencies with the predictor that need to be solved. The error mean shouldn't be considered in an absolute sense but with respect to the actual process values. The most important parameter for evaluating the prediction accuracy is the coefficient of determination  $r^2$ , which is a function of the mean squared error normalized by the variance of the actual data. For a perfect predictor, the coefficient of determination should be 1, whereas for a trivial mean predictor (one that always predicts the mean of the actual data) the coefficient of determination is zero. Although the coefficient of determination can also be negative, it usually doesn't make sense to evaluate a predictor whose  $r^2$  is even close to zero.

#### Step 5

The actual prediction can deal either with predicting a characteristic process parameter for just the next time step (*prediction horizon 1*), or with predicting a parameter several steps ahead (*larger prediction horizon*). Larger prediction horizons are useful in numerous real-life problems like power consumption predictions, car sales predictions, or Internet traffic predictions.

#### Testing the idea

Our experiments tested whether the most appropriate linear stochastic model can provide an indication of the appropriate number of NN inputs:  $p$  external inputs to a feedforward NN for an AR( $p$ ) process, or  $p$  external and  $q$  context inputs to a recurrent NN for an ARMA( $p, q$ ) process. Additionally, they explored whether initial NN weights obtained from the stochastic model as described by Equations 16 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. Since stochastic analysis of the data sets considered in the experiments indicated AR( $p$ ) models as the most appropriate, all the tested NN architectures were of feedforward type.

All of our experiments encompassed a prepro-

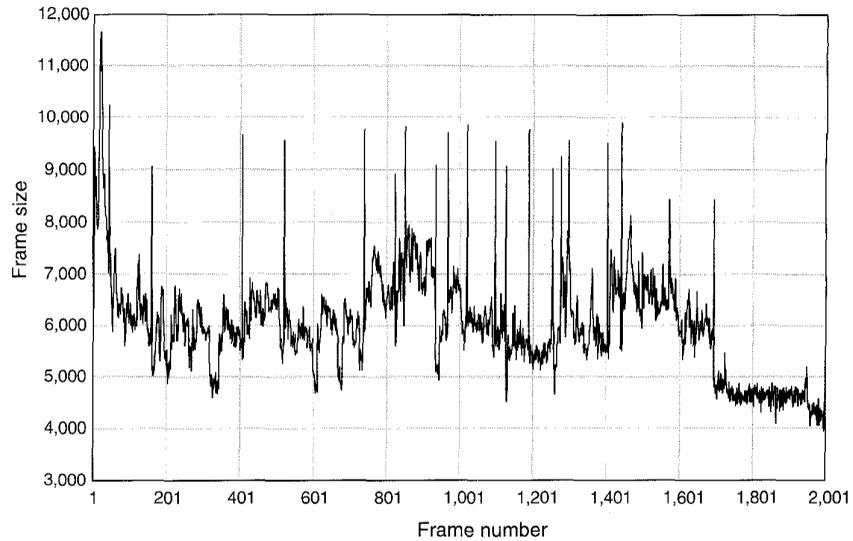


Figure 2. Entertainment video traffic data used in experiment.

cessing, consisting of both a logarithmic smoothing and a first-order differentiation for stationarization purposes. The NN experiments used backpropagation learning with the following parameters: learning rate  $\eta = 0.01$ , momentum term  $\alpha = 0.7$ , tolerance  $t = 0.02$ . Unless stated otherwise, the NN learning encompassed 4,000 passes (epochs) through the training set. We used two very different types of data to test whether prior knowledge derived from stochastic modeling can help in selecting an adequate NN architecture, initial weights, and a sampling rate.

#### Experimental results:

##### Entertainment video traffic data

The first data type used in the experiments consisted of 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.<sup>13</sup> The frame sizes are given in number of cells, each cell having a fixed length of 53 bytes. The difficulty associated with this data set is the *nonstationarity* (data distribution changes over time), as well as the existence of "outliers" (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, it is also important to predict the outliers. The data (see figure 2) considered in our

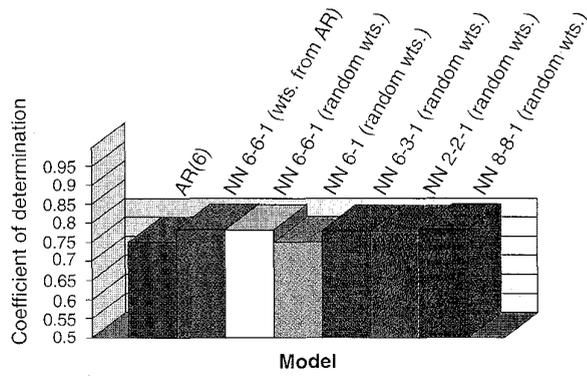


Figure 3. Prediction accuracy for horizon 1 on entertainment video traffic data.

experiments consisted of 2,000 samples, of which the first 1,000 were used for parameter estimation (training) and the last 1,000 for model validation (testing). The actual predictions were done on the same 1,000 samples used for model validation and not on a separate data set.

**Prediction horizon 1**

The NN weights were initialized either with small random values, or from the corresponding AR parameters as in Equations 16. The results presented for the NNs initialized with random weights were averaged over 10 runs. The coefficient of determination for the most appropriate stochastic model, as well as for some NNs, obtained either from the stochastic model or through a trial-and-error procedure, is presented in Figure 3, in which the notation NN  $x-y_1-y_2-z$  stands for an NN with  $x$  external inputs,  $y_1$  and  $y_2$  units in the first and second hidden layers respectively, and  $z$  output units.

The error means were not included in the figure since they indicated that both AR and NN predictors were unbiased (the residual means were negligible compared to the actual data values). As mentioned earlier, the mean of the residuals doesn't serve as a quantitative measure of a predictor's accuracy. It is merely used to signal a deficient predictor when its value relative to the process values is far from zero.

The quantitative evaluator of a predictor's accuracy,  $r^2$ , for different AR( $p$ ) predictors, with  $p$  in the 1-40 range, indicated an AR(6) as the most appropriate stochastic model, thus suggesting the use of a feedforward NN with six inputs. Varying the hidden-layer size suggested that a number of hidden units equal to the number of inputs is an appropriate choice, thus also allow-

ing the NN weight initialization using AR-model parameters. Figure 3 shows a slight improvement in prediction accuracy of the best NN model as compared to the most appropriate AR model. The  $r^2$  values confirmed that the choice of an NN with six inputs, corresponding to the most appropriate AR(6), was adequate (by varying the number of external inputs, hidden neurons, and number of hidden layers, no significant improvement in prediction accuracy was achieved). Whether starting from random weights or by initializing the weights from the AR parameters, the NNs yielded a very similar prediction accuracy. Starting the neural network learning process with weights initialized from the AR parameters could 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 "freezing" of the learning process in a local minimum a small additive noise to the initial weight values could be desirable.

From these experiments we conclude the following:

- ◆ The data set appears to underlie a fairly linear process, so that the improvement obtained by the (nonlinear) NN predictors compared to the (linear) AR models is not significant.
- ◆ The prior knowledge provided by the stochastic analysis of using six input units for NN modeling of the underlying process seems to be appropriate.
- ◆ Mapping the AR parameters onto the NN weights yields a similar prediction accuracy to the case of random weight initialization but speeds up the learning process.

**Increased prediction horizon**

In the previous experiment, the NN did not perform significantly better than the corresponding linear stochastic model. We then experimented with a different prediction objective of increased difficulty, in which we expected the computationally more powerful NN model to yield better performance.<sup>5</sup> This more difficult problem, which is quite important in practice, is prediction for an increased horizon (more than one step ahead in time).

For a larger prediction horizon different sampling rates can be employed, which makes the trial-and-error selection of an NN architecture

even more impractical. Consequently, in this experiment the choice of an appropriate sampling rate based on the stochastic-modeling prior knowledge was also explored. In addition, similar to the prediction horizon 1, we also explored whether an appropriate AR( $p$ ) model indicates the use of a feedforward NN with  $p$  external inputs whose initial weights can be set according to the AR parameters.

The same entertainment video traffic data were used for experimentation, but now with prediction horizon 10 (the 10th-step-ahead process value is predicted). To predict the process at time step  $t + 10$  using  $k$  process values up to time  $t$ , the following uniform sampling rates were considered (all are divisors of the prediction horizon):

- ◆ Rate 1—by ones; the  $k$  previous process values are  $x(t), x(t-1), x(t-2), \dots, x(t-(k-1))$
- ◆ Rate 2—by twos; previous values are  $x(t), x(t-2), x(t-4)$ , etc.
- ◆ Rate 5—by fives; previous values are  $x(t), x(t-5), x(t-10)$ , etc.
- ◆ Rate 10—by tens; previous values are  $x(t), x(t-10), x(t-20)$ , etc.

For horizon  $b$  larger than one, the prediction can be done either in a *direct* or in an *incremental* fashion. In the direct approach, the NN is trained to predict directly the  $b$ th step ahead without predicting any of the intermediate 1, ...,  $b-1$  steps. In the incremental approach, the NN predicts all the intermediate values up to  $b$  steps ahead by using the previously predicted values as inputs when predicting the next value. All NN results were, as indicated, obtained either by initializing the NN weights from the AR parameters, or averaged over 10 runs with different initial random weights.

The most appropriate AR models obtained for different sampling rates, as well as the corresponding NN models, are presented in Figure 4. Again,

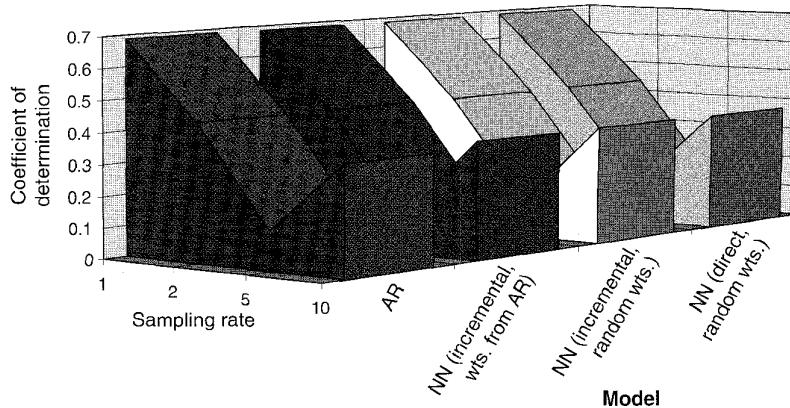


Figure 4. Prediction accuracy for horizon 10 and different sampling rates on entertainment video traffic data.

both stochastic and NN predictors appeared to be unbiased, so the predictors' accuracy was evaluated according to the  $r^2$  values (the higher the  $r^2$  value, the better the predictor). The stochastic models indicated a sampling rate 1 as the most appropriate. The NN results confirmed this. We also observed that the performance of the NNs with weights initialized according to the AR parameters was very similar to that of the NNs averaged over 10 runs with different initial random weights.

Figure 5 summarizes the results obtained for the most appropriate stochastic model, as well as for different representative NNs when using a sampling rate 1. All the NN results were averaged over 10 runs with different initial random weights. The figure indicates that the NN having 11 inputs yielded the best prediction, this being consistent with the prior knowledge provided by the stochastic modeling. We also ex-

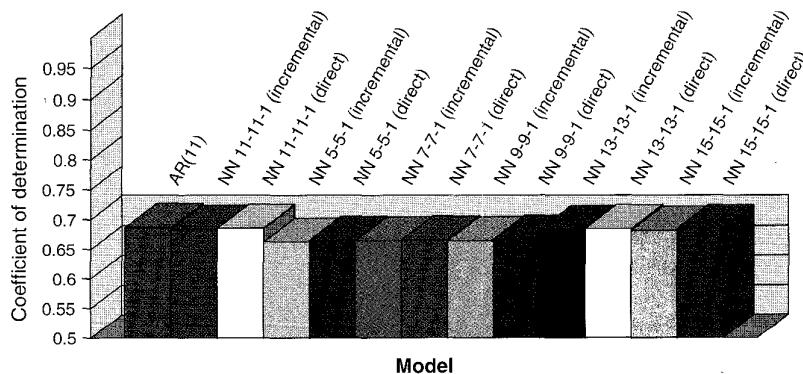


Figure 5. Prediction accuracy for horizon 10 and sampling rate 1 on entertainment video traffic data.

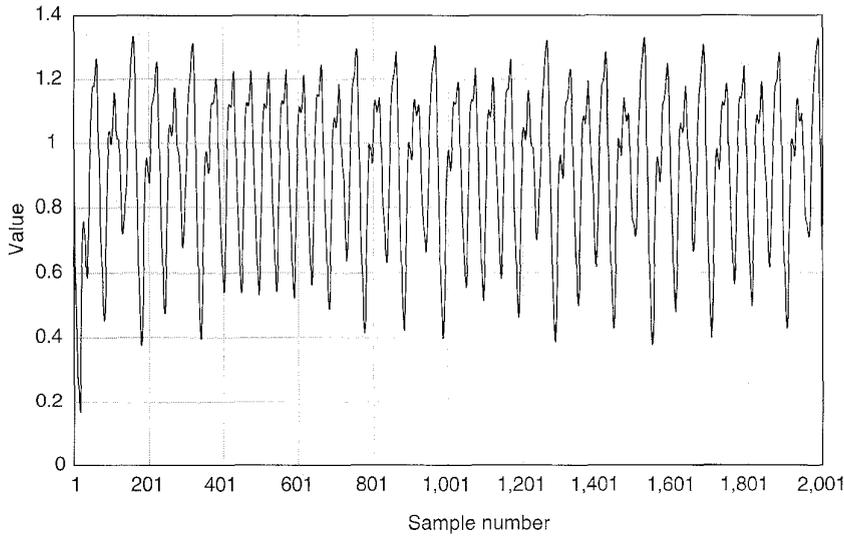


Figure 6. Mackey-Glass data arising from a nonlinear, chaotic time series.

which are shown in Figure 6. 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 with previously published results,<sup>7</sup> a sampling rate 6 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 2,000-sample data set which was used for experimentation on prediction horizons 1 and *k*. The first 1,000 samples of this “filtered” data were used for training, the last 1,000 for testing.

perimented with neural network architectures much larger than the ones indicated, but their performance was poor.

The conclusions that could be drawn from these experiments are:

- ◆ The data-sampling rate indicated by the stochastic models seems to be appropriate also for the NN models.
- ◆ The prior knowledge provided by 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 NNs is comparable, indicating a linearity of the problem.

**Experimental results:  
Mackey-Glass data**

The second data type we used is a deterministic time series obtained by integrating a delay differential equation (also known as the Mackey-Glass series):

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

Experiments were performed for the  $A = 0.2$ ,  $B = 0.1$ ,  $\tau = 17$ , case in which the system exhibits chaotic behavior. The difficulty associated with this data set is the high *nonlinearity*. The data set consisted of 12,000 samples; the first 2,000 of

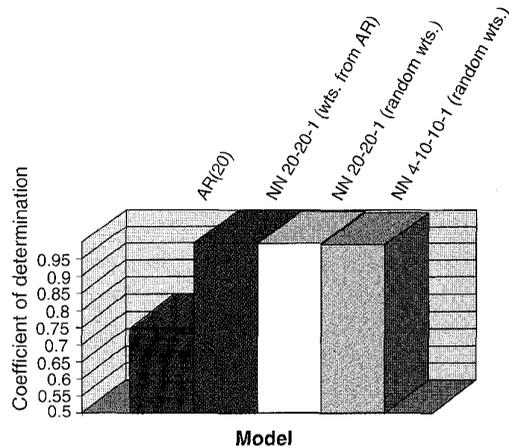
**Prediction horizon 1**

Similar to the previous experiments, the NN results were obtained either as a single run starting with weights derived from the AR model, or as an average over 10 runs with different initial random weights.

Instead of comparing the stochastic-information-based NN to the NNs of somewhat different architectures obtained through a trial-and-error procedure as previously, in these experiments the results were compared with an earlier reported “optimal” NN topology with four inputs and two hidden layers of 10 units each, in which the number of inputs was determined according to Takens’s theorem.<sup>7</sup> The number of training epochs was 2,000 for the NN 20-20-1 architectures and 4,000 for the NN 4-10-10-1 architecture. Both stochastic and NN predictors were unbiased, so the predictors’ accuracy was evaluated according to the  $r^2$  values reported in Figure 7.

The conclusions we draw from this experiment are:

- ◆ The performance of the NNs is much better than that of the most appropriate stochastic model, this being explained by the nonlinearity of the time series.
- ◆ The NN based on stochastic prior knowledge (regarding both the number of inputs and appropriate initial weight values) performed similarly to the “optimal” NN architecture, further supporting the stochastic-informa-



**Figure 7.** Prediction accuracy for horizon 1 on Mackey-Glass data.

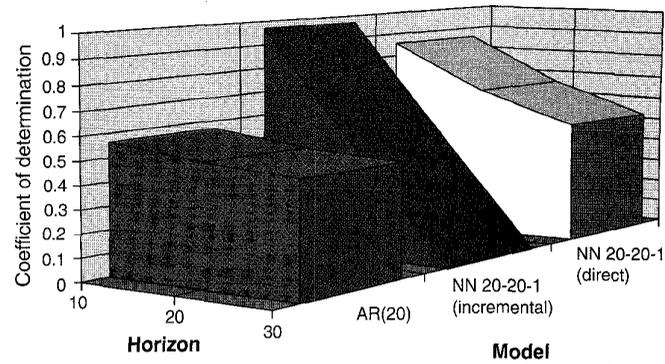
tion-based design approach. This is an important finding, since it confirms that useful information can be extracted from a linear model even if the underlying time series is highly nonlinear.

- ◆ Although the stochastic-information-based NN might appear to be overdimensioned compared to the “optimal” network, training (learning) and prediction in an actual hardware implementation would be faster for the stochastic-information-based architecture since it contains a single layer of hidden units compared to two such layers in the “optimal” architecture.

#### Increased prediction horizon

Whereas corresponding experiments on the entertainment video data were used to predict only 10 steps ahead, for the Mackey-Glass data we analyzed the decrease in prediction accuracy for very large prediction horizons. 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 shown in Figure 6). In the case of the NN, the weights were initialized from the AR parameters. The values for the coefficient of determination,  $r^2$ , resulting from these experiments are presented in Figure 8.

The results indicated that, although the performance of the NN was much better when predicting the near future, it decreased dramatically



**Figure 8.** Prediction accuracy for increased prediction horizon on Mackey-Glass data.

until, after about 30 steps ahead, the NN predictor became completely unusable. (The  $r^2$  values for the NN predictor were 0.969, 0.493 and  $< 0$  for horizons 10, 20, and 30 respectively; for the AR predictor the values were 0.565, 0.515, and 0.488 for horizons 10, 20, and 30 respectively.) This was an indication of the instability of the trained NN (an undesirable accumulation of error when using the incremental approach). For this reason, three 20-20-1 NNs were trained in the *direct* fashion for predicting 10, 20, and 30 steps ahead, respectively. The values for their coefficients of determination (0.858, 0.667, and 0.535 for horizons 10, 20, and 30), obtained as averages over 10 runs with different initial random weights, are also included in Figure 8. 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.

We conclude from this experiment that, although an incremental approach for NN training in the case of an increased prediction horizon has the advantage of training a single NN 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.

Various exploratory analysis techniques have been proposed with the objective of extracting information from a time series that would indicate its predictability, as well as the

appropriateness of a specific prediction method. Although promising diagnostic methods have been reported, one that is general and practical enough to be applied to an arbitrary time series is still unknown. This is generally due to insufficient data for a reliable diagnostic, or complex data interdependencies overlooked by the diagnostic process.

Our study did not attempt to provide a general means of classifying time series into groups that can or cannot be predicted well by certain methods. Instead, the study investigated the possibility of integrating two different prediction methods—stochastic models and neural networks—in order to speed up the design process of an appropriate predictor. In particular, we tested whether feedforward NNs for time series predictions can be rapidly designed by using prior knowledge obtained from stochastic modeling. The generality of this approach was analyzed in the context of two very different time series: a real-life, nonstationary, stochastic time series (the entertainment video traffic data), and an artificially generated, nonlinear, deterministic time series (the Mackey–Glass data).

The existence of a large number of outliers in the entertainment video traffic time series suggests a process that is difficult to predict with high accuracy. This was confirmed in our experiments by a relatively low value of the coefficient of determination for both stochastic and NN models. However, although NNs are computationally more powerful than the linear stochastic models, a simple stochastic model performed comparably to the NNs, justifying the use of linear models for some important real-life problems.

The quasi-periodicity and smoothness of the Mackey–Glass time series suggests a process that is possible to predict with high accuracy if using an appropriate predictor. A high value of the coefficient of determination, significantly larger than for the stochastic predictor, was obtained using an NN, confirming the nonlinearity of the underlying time series. Nevertheless, experiments suggested that linear stochastic analysis provided useful knowledge on selecting the number of NN inputs and initial weights, as well as on choosing an appropriate data-sampling rate.

It is important to emphasize that the goal of the proposed approach was not to find “the optimal” NN architecture for a given problem but to provide rapidly (after a fast stochastic analysis)

an NN architecture with close to optimal performance. Further research is needed to explore the validity of the stochastic prior knowledge to other time series prediction problems, as well as to extend the study from AR to ARMA models (that would indicate the choice of a recurrent NN). We believe that potentially better prediction accuracy and/or more efficient modeling is achievable by integrating prior knowledge and NN learning. For example, a successful integration of expert-system rules and NN classifiers has been demonstrated elsewhere.<sup>14</sup> The approach proposed in this study is a way of incorporating prior knowledge into NN systems for time series prediction. As a further level of integration, our current research considers the use of stochastic modeling with additional sources of prior knowledge (such as information theory and chaotic system analysis) for NN-based time series predictions. ♦

### Acknowledgments

*We thank Paul Werbos for extremely useful discussions at early stages of this work, and George Cybenko and the other reviewers for their critical reading of the manuscript in its final stages. We would also like to thank Steve Elgar for the summer 1995 funding that allowed Radu Drossu to work without interruption at Washington State University during the preparation of this article. Finally, the financial support of NSF grant IRI-9308523 to Zoran Obradovic is gratefully acknowledged.*

### References

1. G. Box, G. Jenkins, and G. Reinsel, *Time Series Analysis: Forecasting and Control*, 3rd ed., Prentice Hall, Englewood Cliffs, N.J., 1994.
2. P. Werbos, “Neural Networks, System Identification and Control in the Chemical Process Industries,” in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, D.A. White and D.A. Sofge, eds., Van Nostrand Reinhold, New York, 1992, pp. 283–356.
3. G. Cybenko, “Approximation by Superpositions of a Sigmoidal Function,” *Mathematics of Control, Signals, and Systems*, Vol. 2, 1989, pp. 303–314.
4. J. Sjöberg et al., “Nonlinear Black-Box Modeling in System Identification: A Unified Overview,” *Automatica*, Vol. 31, No. 12, 1995, pp. 1,691–1,724.
5. Z. Tang, C. de Almeida, and P.A. Fishwick, “Time Series Forecasting Using Neural Networks vs. Box-Jenkins Methodology,” *Simulation*, Vol. 57, No. 5, Nov. 1991, pp. 303–310. Reprinted in *Artificial Neural Networks: Forecasting Time Series*, V.R. Vemuri and R.D. Rogers, eds., IEEE Computer Soc. Press, Los Alamitos, Calif., 1994, pp. 20–27.

6. F. Takens, "Detecting Strange Attractors in Turbulence," in *Dynamical Systems and Turbulence*, D. Rand and L. Young, eds., Lecture Notes in Mathematics 898, Springer-Verlag, Berlin, 1981, pp. 366-381.
7. A. Lapedes and R. Farber, "Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling," Tech. Report LA-UR87-2662, Los Alamos National Laboratory, Los Alamos, N.M., 1987.
8. M.A.S. Potts and D.S. Broomhead, "Time Series Prediction with a Radial Basis Function Neural Network," *Adaptive Signal Processing*, SPIE Proc. Vol. 1565, SPIE, Bellingham, Wash., 1991, pp. 255-266.
9. A. Juditsky et al., "Nonlinear Black-Box Models in System Identification: Mathematical Foundations," *Automatica*, Vol. 31, No. 12, 1995, pp. 1,725-1,750.
10. J.T. Connor et al., "Recurrent Neural Networks and Robust Time Series Prediction," *IEEE Trans. Neural Networks*, Vol. 5, No. 2, Mar. 1994, pp. 240-254.
11. P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, doctoral dissertation, Harvard Univ., Cambridge, Mass., 1974. Reprinted as P. Werbos, *The Roots of Back-propagation: From Ordered Derivatives to Neural Networks and Political Forecasting*, John Wiley & Sons, New York, 1994.
12. S.M. Kay, *Modern Spectral Estimation: Theory and Application*, Prentice Hall, Englewood Cliffs, N.J., 1988.
13. R. Drossu et al., "Single and Multiple Frame Video Traffic Prediction Using Neural Network Models," in *Computer Networks, Architecture and Applications*, S.V. Raghavan and B.N. Jain, eds., Chapman and Hall, New York, 1995, pp. 146-158.
14. J. Fletcher and Z. Obradovic, "Combining Prior Symbolic Knowledge and Constructive Neural Networks," *Connection Science: Journal of Neural Computing, Artificial Intelligence, and Cognitive Research*, Vol. 5, Nos. 3-4, 1993, pp. 365-375.

**Radu Drossu** received his MS degree in electrical engineering from the Polytechnical Institute of Bucharest, Romania, in 1990. He worked as a system programmer and hardware designer at the Research Institute for Automation (IPA), Bucharest, from 1990 to 1993. He is currently a PhD candidate in computer science at Washington State University, doing his research in artificial neural networks under the supervision of Zoran Obradovic.

**Zoran Obradovic** received a BS in applied mathematics, information and computer sciences in 1985 and an MS in mathematics and computer science in 1987, both from the University of Belgrade, and a PhD in computer science from the Pennsylvania State University in 1991. He is currently a research scientist at the Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade, and an assistant professor in the School of Electrical Engineering and Computer Science, Washington State University. His research explores the applicability of neural network technology to large-scale classification and time series prediction problems in very noisy domains.

Readers may contact the authors in care of Z. Obradovic, School of Electrical Engineering and Computer Science, Washington State Univ., Pullman, WA 99164-2752; e-mail, {zoran,rdrossu}@eecs.wsu.edu.





**Distributed  
Objects  
Methodologies for  
Customizing  
Systems Software**  
by Nayeem Islam

Companies that build software are now focusing on building application-specific systems software. This is a difficult task that many researchers have been trying to cope with for a number of years. To help solve such problems, the author presents a new approach to designing customized system software that is application-specific and based on object-oriented frameworks. The technology presented has been influential in the design and implementation of several products at Microsoft, IBM and SUN Microsystems.

275 pages. March 1996. Hardcover. ISBN 0-8186-7193-9.  
Catalog # BP07193 — \$39.00 Members / \$45.00 List

50 YEARS OF SERVICE

IEEE

**COMPUTER  
SOCIETY** 

1946-1996

Call toll-free:  
**+1-800-CS-BOOKS**

FAX Orders:  
**+1-714-821-4641**