RESEARCH NOTE

SIGNAL PREDICTION BY A LAYERED FEED-FORWARD NEURAL NETWORK

A. Khotanzad and J.H. Lu
Electrical Engineering Department, Southern Methodist University
Dallas, Texas U.S.A

Abstract In this paper a nonparametric neural network (NN) technique for prediction of future values of a signal based on its past history is presented. This approach bypasses modeling, identification, and parameter estimation phases that are required by conventional parametric techniques. A multi-layer feed forward NN is employed. It develops an internal model of the signal through a training operation involving past history of the considered signal. Training is performed using the back-propagation algorithm. The trained net is then used to do the forecast. Training is continued during operation to improve performance. The net performance is tested on signals generated by autoregressive (AR) and autoregressive moving-average (ARMA) models of different orders and results are compared to optimal forecasts.

چکیده دراین مقاله، یک تکنیک شبکه عصبی غیر پارامتری، جهت تخمین مقادیر آتی یک سیگنال براساس مقادیر گذشته آن پیشنهاد گردیده است. این روش، فازهای مدل سازی، تعیین هویت، تخمین پارامتری که در تکنیک‌های معمولی مورد نیاز شده است را کنار میگذارد. یک شبکه عصبی چند لایه‌ای NN بکار گرفته شده است. مدل داخلی سیگنال با تعلیم شبکه براساس مقادیر گذشتهٔ سیگنال تشکیل میگردد. تعلیم شبکه براساس الگوریتم انتقال به عقب انجام میگیرد. شبکه تعلیم شده سپس جهت پیشگویی سیگنال بکار برده شده است. تعلیم شبکه همزمان با عملیات تخمین جهت بالابردن کارایی سیستم ادامه مییابد. کارایی شبکه با بکارگیری سیگنالهائی که از طریق مدلهای AR و ARMA با مرتبه‌های مختلف تولید شده‌اند امتحان گردیده و نتایج با پیشگویی‌های بهینه مقایسه گردیده‌اند.

INTRODUCTION

Signal prediction from past observations has applications in many diverse fields such as target tracking, weather forecasting, power system loading forecasting, financial market forecasting, etc. Conventional parametric approaches to this problem involve mathematical modeling of the signal characteristics which is then used to carry out the prediction. In a general case, this is a rather complex task involving many steps such as model hypothesis (commonly a time series), identification and estimation of model parameters and its verification. In many instances, adaptive update of the estimated parameters is also necessary.

In this study, a nonparametric neural network-based approach to the signal prediction problem is presented. The method is non-parametric in the sense that it does not need to know any information regarding the process that generates the signal. For instance the order and parameters of an AR process are not needed in order to carry out the prediction. This reflects a significant difference between this method and the conventional ones, since it bypasses the mentioned modeling phases. Although signals generated from autoregressive (AR) and autoregressive moving-average (ARMA) processes will be used in this study, the technique is not restrictive and can be applied to other types of signals.

A NN is composed of many simple and similar nonlinear computational elements (nodes) operating in parallel and arranged in patterns reminiscent of biological nervous systems. The nodes are densely interconnected via weights that can adapt and change according to a given situation. In a typical neural network, at each node, the inputs from the other nodes are multiplied by suitable weights, summed and the result transformed by a no-memory nonlinearity transfer function. The nodes are often grouped together into linear arrays called layers. The network usually consists of an input layer to which data is applied, an output layer and one or more "hidden" layers, so-called because neither their inputs nor outputs are available to the outside world. If all the connections in the network are from a previous layer to the next layer, the network is a "feed-forward" network.

The main property of a NN such that is utilized in this

study is the ability to learn complex mappings between input and output vectors which are very difficult to embody in conventional algorithmic methods. This task is carried out by a process of learning from examples presented to the net. During learning, known input-output pairs, called the training set, are applied to the network. The network learns by adjusting or adapting the strengths of the connections between processing elements, by comparing the actual output of the net to the expected output. The method used for doing the adaptation is called the learning or adaptation rule.

In this case, the learning consists of showing the net examples of past signal behavior. After the net learns, it is employed to track the signal. Furthermore, a continuous learning is built into the system which allows the net to update its weights on-line proportional to its forecast error.

THE NET AND ITS LEARNING RULE

In this study, we use a multi-layer feed-forward neural network topology with one hidden-layer as shown in Figure 1. The input nodes (shown with filled circles) have a transfer function of unity and do not alter the input to them. They serve as the distributors of inputs to the first hidden layer nodes through the corresponding connections. The nonlinear activation function of all the other nodes is the sigmoid function, i.e. X_j, the output of node j is:

$$X_j = \frac{1}{+\exp(-\sum_i w_{ij} X_i)}$$

where w_{ij} is the strength of connection between node j and node i in the layer below.

The learning process to be utilized is an iterative gradient procedure referred to as "Back-Propagation Algorithm" [5]. According to this algorithm which is next outlined, for each input in the training set, learning proper weights is conducted by (1) computing the discrepancy between the desired outputs and that which is produced by the network outputs, and (2) feeding back this error signal level-by-level to the inputs, changing the connection weights in such a way as to modify them in proportion to their responsibility for the output error. The major steps of the algorithm are as follows:

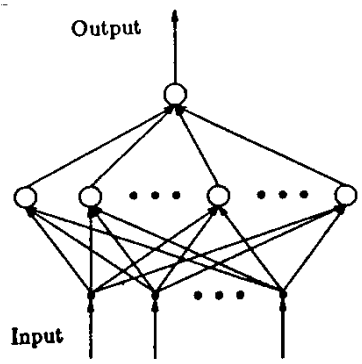

Figure 1. The multi-layer feed-forward neural network utilized in this study

Step 1 Initialize all w_{ij}s to small random values (typically between -0.5 to 0.5).

Step 2 Present an input vector and specify the corresponding desired outputs.

Step 3 Calculate outputs using the present value of w_{ij}s.

Step 4 Find an error term, δ_j, for all the nodes. If d_j, O_j and X_j stand for desired value of the jth output node, the computed value for the jth output node, and the computed value for the jth hidden layer node then error for an output node is:
$\delta_j = (d_j - O_j) O_j (1 - O_j)$,
and for a hidden layer node is:
$\delta_j = X_j (1 - X_j) \sum_k \delta_k w_{jk}$, where k is over all nodes in the layer above node j.

Step 5 Adjust weights by
$w_{ij}(n+1) = w_{ij}(n) + \alpha \delta_j X_i + \zeta (w_{ij}(n) - w_{ij}(n-1))$,
where (n+1), (n), and (n-1) index next, present, and previous, respectively. α is a learning rate similar to step size in gradient search algorithms. ζ is a constant between 0 and 1 which determines the effect of past weight changes on the current direction of movement in weight space.

Step 6 Present another input and go back to step 2. All the training inputs are presented cyclically until weights stabilize.

A big drawback of the back-propagation algorithm is its slow rate of convergence. Typically, many passes through the training data are required for the connection weights to converge. Consequently, it may take up to

TABLE 1. Illustration of Data Segmentation for the Initial Training Set.

m-Point Input	Desired Output
y(1),..., y(m)	y(m+1)
y(2),..., y(m+1)	y(m+2)
y(3),..., y(m+2)	y(m+3)
.	.
.	..
.	.
y(k),..., y(k+m-1)	y(k+m)

TABLE 2. Prediction Error Statistics for Signals Generated by AR Processes

Order of AR process used to generate the signal	Neural Net		Linear Prediction	
	Error Mean	Error STD	Error Mean	Error STD
2	.441	.113	.403	.093
3	.417	.099	.398	.091
4	.453	.131	.388	.087
5	.414	.098	.395	.088
6	.437	.112	.392	.090
7	.431	.105	.403	.089
8	.433	.110	.404	.095
9	.421	.102	.406	.094
10	.415	.096	.402	.090

several hours to train on a serial computer. However, the structure of the network and the algorithm are very well suited to parallel implementation which will cut down the amount of required time. The above parallelization has been implemented on our SEQUENT Symmetry S81parallel computer. This is a Multi Instruction Multi Data (MIMD) Shared Memory parallel computer with 20 processors, each capable of performing 4 MIPs (million instructions per second). Although our data partitioning is limited to $L \leq 20$, the reduction in the training time is enough to make use of the network practical.

In recent years, this kind of NN has been applied to a number of pattern classification problems [3]. However, in all of these applications, the network has been used in the context of a classifier: a quite different function from what it will be utilized for in this research. In [1], the authors have shown that a NN classifier applied to the image recognition problem outperforms all of the other traditional classifiers, indicating the importance and potential of the neural network technology.

DESCRIPTION OF THE APPROACH

To illustrate the procedure, assume that several data points (observations) denoted by $y(1), y(2), \ldots, y(n)$ are available. Let us concentrate on one-step-ahead prediction of this process based on the past m observations ($m<<n$). The generated data are divided into overlapping sections of length m+1. Each section overlaps the preceding section by m points. These sections compromise the initial training set for the net. The first m points of each section are used as inputs to the net and the (m+1)th is designated as the corresponding desired output.

Table 1 illustrates this arrangement assuming that with using k such a section can be made out of the available data.

The net is trained off-line with such input and output pairs. Then it is used for on-line forecasting. It is assumed that the actual observation becomes available after its forecast is made. Therefore one can compute the forecast error as

$$\varepsilon(j) = y(j) - \hat{y}(j)$$

where $y(j)$ is the actual observation and $\hat{y}(j)$ is its forecast.

One can take advantage of the observed error to further improve the initial training of the net. Thus the training continues during the testing/operation stage. This means that after prediction error $\varepsilon(j)$ is computed, it is propagated back using back-propagation to adjust the weights again before attempting the prediction of the next point. This is of course based on the assumption that actual observation becomes available before the next step prediction needs to take place. Thus the net is not statistic and its weights are continuously adjusted according to the most recent observation.

The predicted value is to be available at the output of the NN. However, note that the sigmoid function is bounded between 0 and 1 while the signal may have a different dynamic range. To compensate for this factor, the data are first normalized by linearly mapping them to [0,1] range. This is accomplished by establishing a minimum and a maximum value for the process. The normalized value of a data point y, denoted by y_N, will then be

$$y_N = \frac{y - y_{min}}{y_{max} - y_{min}}$$

which is between 0 and 1. The output of the net (the forecast), which will also be between 0 and 1, can be scaled back up through inverse mapping. Thus, the net will operate on [0,1] range data values and sigmoid functions. Note that this is a linear mapping which does not alter the characteristic of the data. Therefore, selection of accurate extrema values is not critical to the success of the procedure.

The described procedure is by no means restricted to one step-ahead prediction. By increasing the output nodes to j, and using sections of length (m+j), one can perform one-to-j-setp ahead prediction.

EXPERIMENTAL STUDY

Several experiments with signals generated by AR and ARMA processes are carried out. In an AR model of order p (AR(p)), the value of the signal at time j, y(j), is expressed as a linear combination of p previous values plus a random noise term $\omega(j)$, that is

$$y(j) = \sum_{i=1}^{p} \theta_i y(j-i) + \omega(j)$$

The noise is white with zero mean and variance of σ^2.

AR processes with orders ranging from 2 to 10 are considered. In addition, four different models (signals) per order are generated. This means that four different sets of θ_i are used for each p, where p=2, ... , 10. The variance of the noise is fixed at $\sigma^2 = 0.25$ for all cases. 2000 points are generated for each series.

A NN with one hidden layer containing 20 nodes is used. In each case, the forecast is based on the past ten observations regardless of the order of the considered process. Therefore the m parameter is m =10. Out of the 2000 generated points, 500 are used for training and the remaining 1500 for testing.

The selected parameters for the back-propagation learning algorithm are: α(learning rate) = 0.2, ζ (momentum) = 0.7, and the number of passes over the training set is 1000. A slight modification to the NN is also made in the form of changing the slope of the output node sigmoid function so that it approximates a ramp function. It was found that such a sigmode performs better. Thus the power of the exponential term in the sigmoid expression is divided by 5 to decrease its slope and make it follow a ramp function.

The performance of the NN is to be compared to that of the linear prediction (LP) which is E[y(j)] expressed as

$$E[y(j)] = \sum_{i=1}^{p} \theta_i y(j-i)$$

Exact LP computation is possible if exact values of p and θ_i s are available. However, in practice these parameters are unknown and need to be estimated from past observations [4]. In this study these parameters are not estimated and exact values are used for LP. Thus the reported LP forecasts are the optimal ones. The LP absolute error is computed as

$$|y(j) - E[y(j)]| = |\omega(j)| \quad k = 501, 502, \ldots, 2000$$

Since $\omega(j) \sim N(0, .25)$, the LP error mean and standard deviation will be .399 and .091, respectively.

Table 2 shows the forecast error statistics for both the NN approach and linear prediction averaged over four cases per order. As one can see from this table the means and variances of prediction errors of these two methods are very comparable indicating the success of the NN method.

In another set of experiments, signals generated by ARMA processes are considered. An ARMA(p,q) process is described by

$$y(j) = \sum_{i=1}^{p} \theta_i y(j-i) + \sum_{i=1}^{q} \alpha_i \omega(j-i) + \omega(j)$$

The noise is white with zero mean and variance of one. 1000 points are generated for each considered signal. 500 of them are used for training and 500 for testing. Figures 2 to 4 show the results in graphical form. Again note that the NN is quite successful in tracking the signal.

It is significant that such good results are obtained without using any knowledge about either the order or parameters of the underlying processes that are used to generate the signal. The net codes the appropriate process through training and retraining during the operation phase.

CONCLUSION

In this study a NN based technique for signal prediction is developed. It is a non-parametric method that does not require any explicit information about the mathematical form of the signal. Although the data sets used in this experiment were generated by AR and ARMA processes with different orders and parameters, the NN based technique worked equally well for all of them. This is a big advantage over the traditional techniques. Comparisons with the optimal forecast show that the NN performance is quite satisfactory.

ACKNOWLEDGMENT

The ARMA experiments were performed by Mr. Mark Fowler. His contribution is gratefully acknowledged.

REFERENCES

1. A. Khotanzad and J.H.Lu, "Classification of Invariant Image Representations Using a Neural Network," to appear in *IEEE Transactions on Acoustics, Speech, and Signal Processing.*, 38, (1990).
2. A. Khotanzad, and J.H.Lu, "Non-Parametric Prediction of AR Processes Using Neural Networks," Int. conf. Acous., Speech, Signal Proc., ICASSP-90, pp. 2551-2554, Albuquerque, NM, April, (1990).
3. R.P. Lippmann, "An Introduction to Computing with Neural Nets, "*IEEE ASSP Magazine*, Vol. 4, No. 2, PP. 4-22, April (1987).
4. S.L. Marple, Digital Spectral Analysis with Applications, Prentice-Hell, New Jersey, (1987).
5. D.E. Rumelhart, and J.L. McClelland, Parallel Distributed Processing:, Vol. 1: Foundations. MIT press, MA., (1986).
6. P. Werbos, "Learning How the World Works: Specifications for Predictive Networks in Robots and Brains, "Proceedings of the 1987 IEEE Int. Conf. on SMC, Vol.1 (1987).

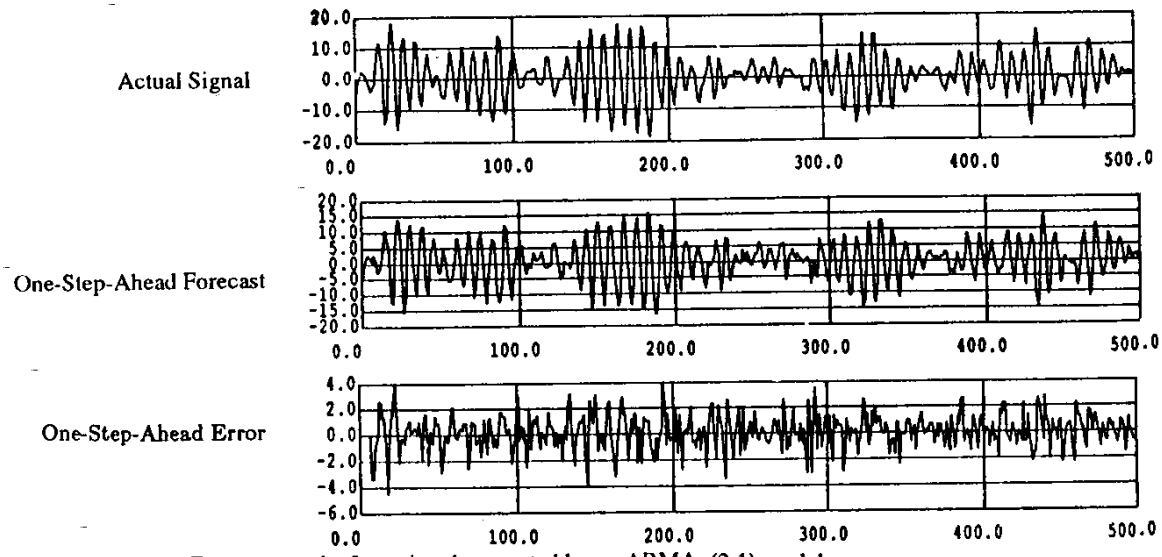

Figure 2. Forecast results for a signal generated by an ARMA .(2,1) model.

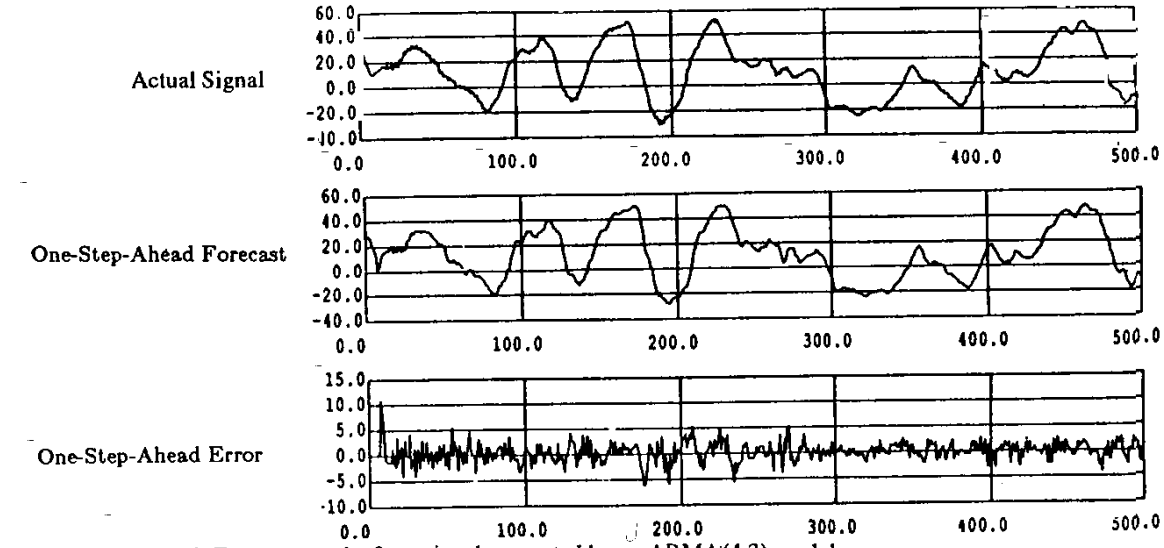

Figure 3. Forecast results for a signal generated by an ARMA(4,3) model.

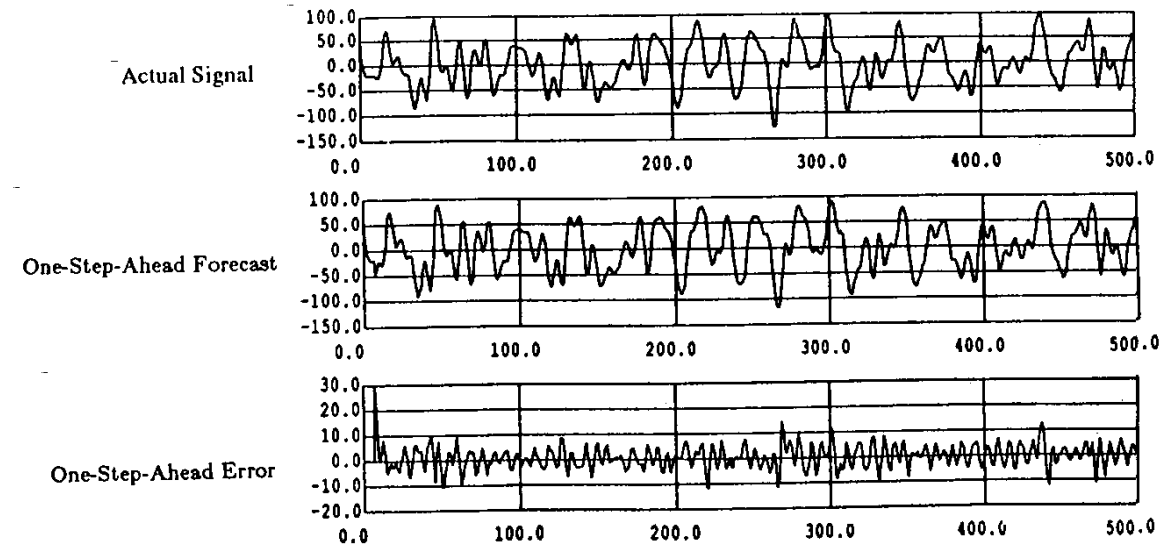

Figure 4. Forecast results for a signal generated by an ARMA(5,4) model.