11-1990

Forecasting the Behavior of Multivariate Time Series using Neural Networks

Kanad Charkraborty

Kishan Mehrotra
Syracuse University, mehrtra@syr.edu

Chilukuri K. Mohan
Syracuse University, ckmohan@syr.edu

Sanjay Ranka
Syracuse University

SU-CIS-90-36

Forecasting the Behavior of Multivariate Time Series using Neural Networks

Kanad Charkraborty, Kishan Mehrotra, Chilukuri K. Mohan, Sanjay Ranka

November 1990

School of Computer and Information Science
Suite 4-116
Center for Science and Technology
Syracuse, New York 13244-4100

(315) 443-2368

SU-CIS-90-36

Forecasting the Behavior of Multivariate Time Series using Neural Networks

K. Charkraborty, K. Mehrotra, C. Mohan, and S. Ranka

November 1990

School of Computer and Information Science
Syracuse University
Suite 4-116, Center for Science and Technology
Syracuse, New York 13244-4100

Forecasting the Behavior of Multivariate Time Series using Neural Networks

Kanad Chakraborty, Kishan Mehrotra, Chilukuri K. Mohan, Sanjay Ranka
4-116 Center for Science and Technology,
School of Computer and Information Science,
Syracuse University, Syracuse, NY 13244-4100
315-443-2368
(e-mail : kanad/kishan/mohan/ranka@top.cis.syr.edu)

December 4, 1990

Abstract

This paper presents a neural network approach to multivariate time-series analysis. Real world observations of flour prices in three cities have been used as a benchmark in our experiments. Feedforward connectionist networks have been designed to model flour prices over the period from August 1972 to November 1980 for the cities of Buffalo, Minneapolis, and Kansas City. Remarkable success has been achieved in training the networks to learn the price curve for each of these cities, and thereby to make accurate price predictions. Our results show that the neural network approach leads to better predictions than the autoregressive moving average(ARMA) model of Tiao and Tsay [TiTs 89]. Our method is not problem-specific, and can be applied to other problems in the fields of dynamical system modeling, recognition, prediction and control.

Key words and phrases: Neural networks, Multivariate time series, Autoregressive moving average models, Prediction.

1 Introduction

Predicting the future is the prime motivation behind the search for laws that explain certain phenomena. As observed by Weigend *et al.* [WAHR 90], it hinges on two types of knowledge: knowledge of underlying laws, a very powerful and accurate means of prediction, and the discovery of strong empirical regularities in observations of a given system. However, there are problems with both approaches – discovery of laws underlying the behavior of a system is often a difficult task, and empirical regularities or periodicities are not always evident, and can often be masked by noise.

Multivariate time-series analysis is an important statistical tool to study the behavior of time dependent data, and forecast future values depending on the history of variations in the data. A time-series is a sequence of values measured over time, in discrete or continuous time units. By studying many related variables together than by studying just one, a better understanding is often obtained. A multivariate time-series consists of sequences of values of several contemporaneous variables changing with time. An important case is when the variables being measured are significantly correlated, *e.g.*, when similar attributes are being measured at different geographic locations. In forecasting new values for each variable, better prediction capabilities are available if variations in the other variables are also taken into account. Robust forecasting must rely on all available correlations and empirical interdependencies among different temporal sequences.

Many past time-series analysis techniques assume linear relationships among variables [BoJe 70]. But in the real world, temporal variations in data do not exhibit simple regularities, and are difficult to analyze and predict accurately. Linear recurrence relations and their combinations for describing the behavior of such data are often found to be inadequate. It seems necessary, therefore, that non-linear models be used for the analysis of real-world temporal data. But formulation of reasonable non-linear models is an extremely difficult task, because of simplifications made in the modeling stage, *e.g.*, omitting parameters which are unknown or which do not seem to affect the observed data directly. Also, the relationships between known parameters and observed values can only be hypothesized, with no simple laws governing their mutual behavior. Hence we resort to a 'neural network' approach for non-linear modeling of multivariate time series; in earlier work, we have successfully used this approach in analyzing univariate time series [LMMR 90].

Neural networks belong to the class of *data-driven* approaches, as opposed to model-driven approaches. The analysis depends on available data, with little rationalization about possible interactions. Relationships between variables, models, laws and predictions are

constructed post-facto after building a machine whose behavior simulates the data being studied. The process of constructing such a machine based on available data is addressed by certain general-purpose algorithms like 'back-propagation' [RuHW 86].

In this paper, we use neural networks to predict future values of possibly noisy multivariate time series based on past histories. The particular data analyzed are monthly flour prices for Buffalo, Minneapolis and Kansas City over a period of a hundred months. For impartial evaluation of the prediction performance of the approach, data for different periods are used in the 'training' (modeling) and 'testing' (prediction) phases. The performance exceeded expectations, and the root mean squared errors (in prediction) obtained using this approach are better than those obtained from the statistical model by at least an order of magnitude. We expect such results to be obtained in other applications as well, since no specific domain knowledge or expertise was used to tune the performance of the neural network.

Section 2 presents the architecture of the neural network used for our analysis, the experiments performed, and the training paradigm used. In section 3, a traditional 'autoregressive moving average' (ARMA) model of statistical prediction has been described, and its performance compared in section 4 with the network performance. Discussion and concluding remarks then follow.

2 Methodology

2.1 Neural Networks

(Artificial) Neural networks are computing systems containing many simple non-linear computing units or nodes interconnected by links. In a 'feedforward' network, the units can be partitioned into layers, with links from each unit in the k^{th} layer being directed (only) to each unit in the $(k+1)^{th}$ layer. Inputs from the environment enter the first layer, and outputs from the network are manifested at the last layer. A $d-n-1$ network, shown in Figure 1, refers to a network with d inputs, n units in the intermediate 'hidden' layer, and one unit in the output layer [WAHR 90]. A weight or 'connection strength' is associated with each link, and a network 'learns' or is trained by modifying these weights, thereby modifying the network function which maps inputs to outputs.

We use such $d-n-1$ networks to learn and then predict the behavior of multivariate

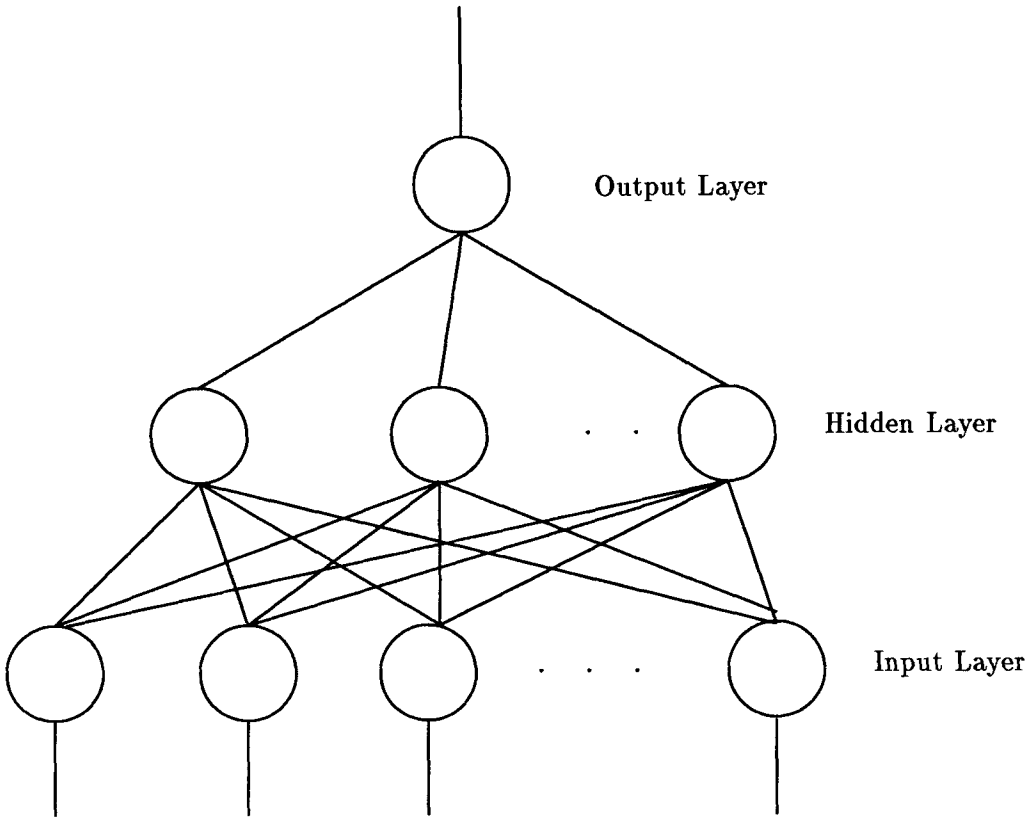

Figure 1: A feedforward neural net with one hidden layer

time series. The hidden and output nodes realize non-linear functions of the form

$$f(x_1, \ldots, x_m) = \frac{1}{1 + e^{-\sum_{1 \le i \le m} w_i x_i + \theta}},$$

where w_i's denote real-valued weights of edges incident on a node, θ denotes the adjustable 'threshold' for that node, and m denotes the number of inputs to the node from the previous layer.

2.2 Experiments

In our experiments, we have analyzed a trivariate time series $X_T = \{(x_t, y_t, z_t) : t = 1, 2, \ldots, T\}$, where T ranges upto 100. The data used are logarithms of the indices of

monthly flour prices for Buffalo (x_t), Minneapolis (y_t) and Kansas City (z_t), over the period from August 1972 to November 1980, obtained from [TiTs 89]. In all cases, we train the network over a certain part of our data, and once training is completed, "test" the network over the remaining data – *i.e.* make the network predict the socalled "future" values.

Both one-lag and multi-lag output predictions are done for the given models. In one-lag prediction, we forecast flour prices of each year based on actual past values only. In multi-lag prediction, on the other hand, we append the predicted values to our input database and use these values also to predict future values. For instance, if the network is used to predict a value n_6 from observed input data $i_1, \ldots i_5$, then the next network prediction n_7 is made using inputs $i_2, \ldots, i_5, n_6$, and the subsequent network prediction n_8 is made using inputs i_3, i_4, i_5, n_6, n_7. With one-lag prediction, on the other hand, the prediction at the eighth instant is made using only the actual input data values i_3, i_4, i_5, i_6, i_7. The following three sets of experiments were performed in this study.

1. Separate Modeling : Each univariate time series $x_T = \{x_t : t = 1, 2, \ldots, T\}$, $y_T = \{y_t : t = 1, 2, \ldots, T\}$, and $z_T = \{z_t : t = 1, 2, \ldots, T\}$, was analyzed separately, without utilizing their interdependencies. For example, only the values of $x_1, \ldots, x_k$ were used to predict x_{k+1}. A separate neural network was used for each of the three series, as illustrated in Figure 2 and trained with about 90 input data values, ranging from August 1972 to January 1980. The training phase is followed by output prediction for the next ten time points (for February 1980 to November 1980) using the weights and thresholds generated during training. These predictions were compared with the test data set to judge the performance of the network. Experiments were performed with *2-2-1, 4-4-1, 6-6-1* and *8-8-1* networks. The learning and prediction capabilities of the networks were found to be poor, and consequently, the separate modeling and prediction approach was abandoned in favor of combined modeling, described below.

2. Combined Modeling : We obtained vastly improved performance using (for each series) information from all series, instead of treating each series in isolation. This is illustrated in Figure 3, in which x_{t+1} is shown as being learned/predicted using six preceding values from all the three series. Similar diagrams can be drawn for the y_{t+1} and z_{t+1} also. For instance, previous x,y and z values were used in predicting a new z value. Furthermore, for the data studied, there was an implicit ordering between the three series: x_t values were available before y_t values, and y_t values were available before z_t values, and (naturally) all these were available before x_{t+1} values. So in

the neural network corresponding to each series, inputs reflected the past histories of that series as well as the others. For instance, in the $d - n - 1$ feedforward network used to predict y_t, if $d = 5$, the chosen input values would be $x_t, z_{t-1}, y_{t-1}, x_{t-1}, z_{t-2}$. As in the previous case, the training set consisted of the first 90 items of trivariate data, and the results shown in Figures 4 through 21 compare performance of an *8-8-1* network with that of a classical autoregressive moving average (ARMA) model. In all the graphs shown, the y-axis is labeled "LFPI", an abbreviation for "*L*ogarithms of monthly *F*lour *P*rice *I*ndices".

3. **Single Modeling** : Success of the above experiments suggested the use of one single neural network (with only one set of weights) to learn all three series together, as shown in Figure 22. The trivariate series was reformulated as a (longer) univariate time series $\{x_1, y_1, z_1, x_2, y_2, z_2, \ldots\}$, and the generic name $\{u_t\}$ was given to this series. Here, the training set consisted of the first 270 observations of the reformulated series. However, this model performed poorly with respect to both training and prediction, and was consequently rejected. The poor performance in this case indicates that the superposition of three time series is much more difficult to learn than a single one. Perhaps the reason is that in using the same set of weights to predict all three series, there is an implicit erroneous assumption that all three series are expected to fit the same model.

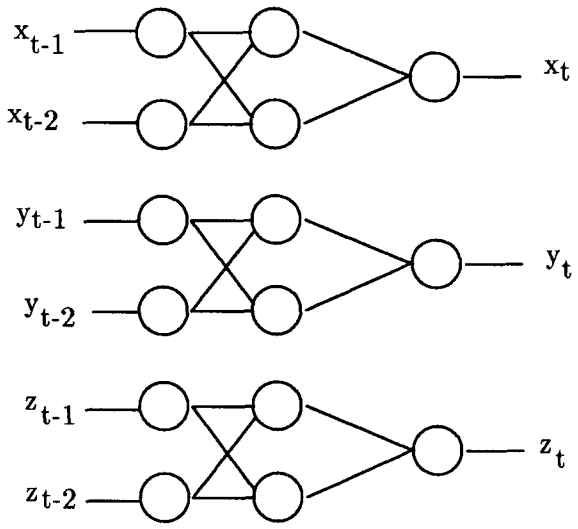

Figure 2: Separate Architectures Schema

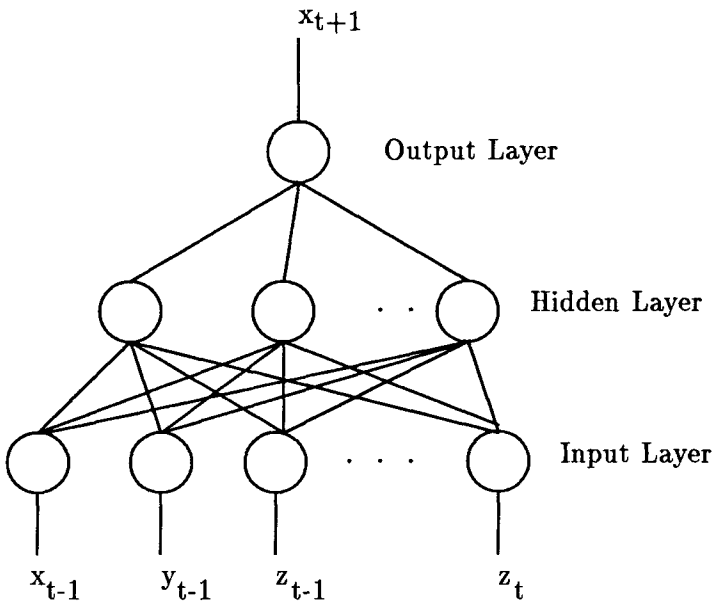

Figure 3: Combined Architectures Schema

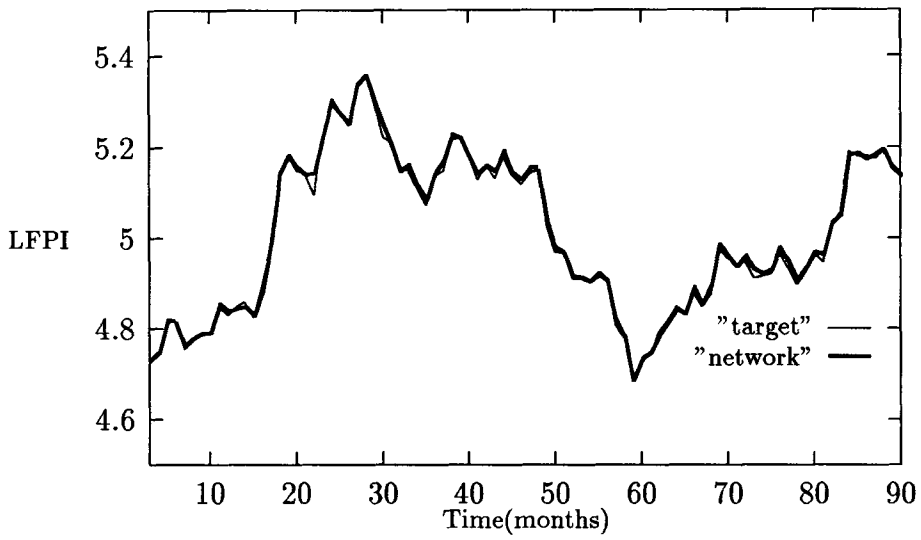

Figure 4: Combined Network Modeling: Buffalo (training)

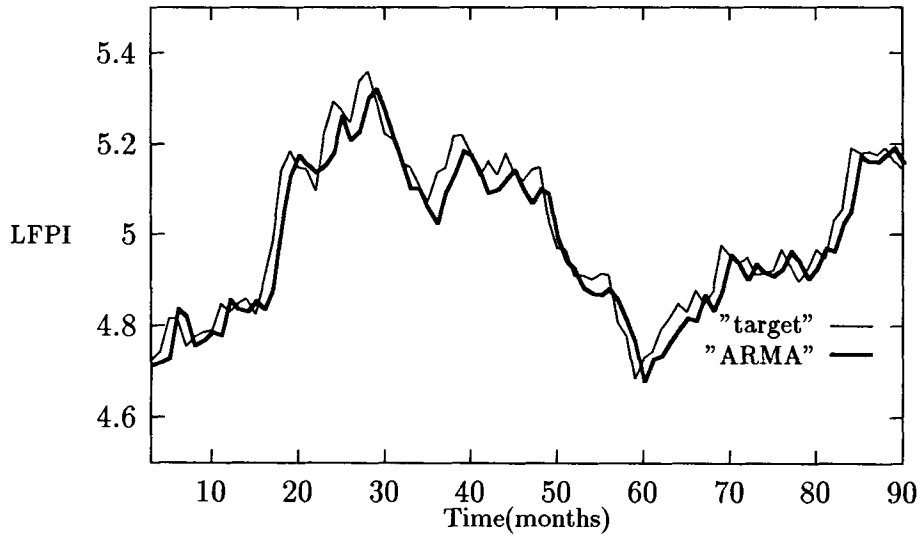

Figure 5: ARMA Modeling: Buffalo (90 months)

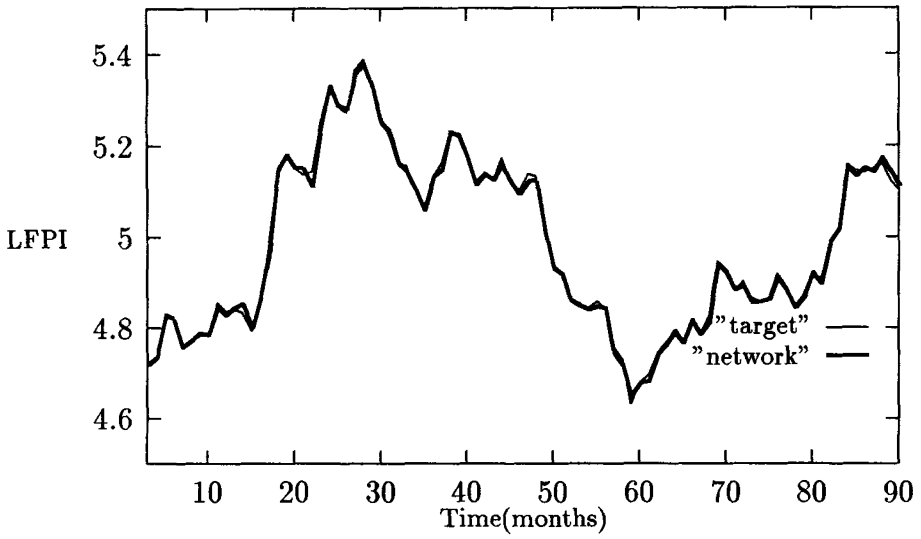

Figure 6: Combined Network Modeling: Minneapolis (training)

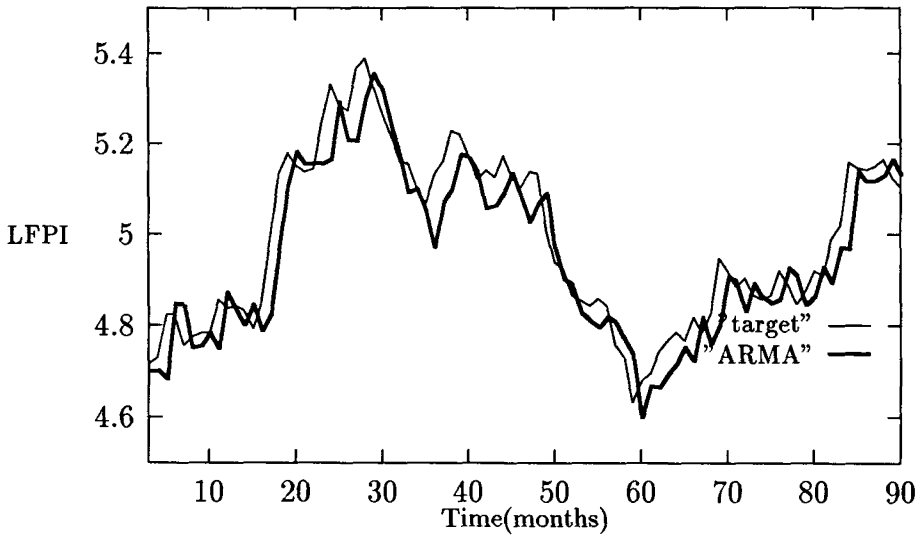

Figure 7: ARMA Modeling: Minneapolis (90 months)

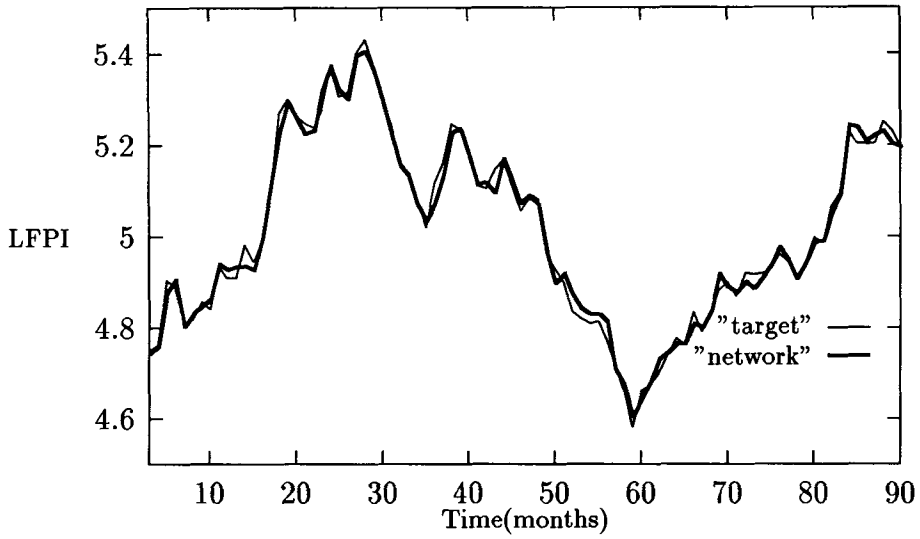

Figure 8: Combined Network Modeling: Kansas City (training)

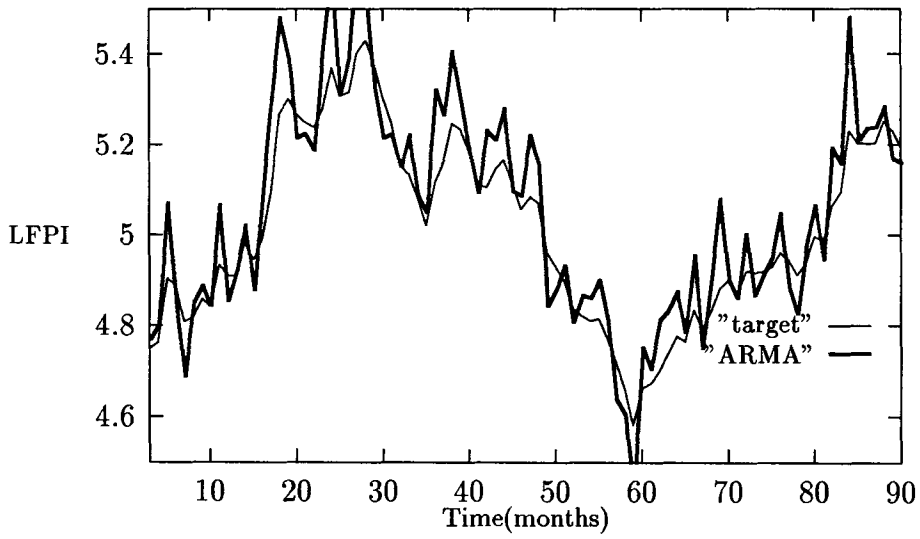

Figure 9: ARMA Modeling: Kansas City (90 months)

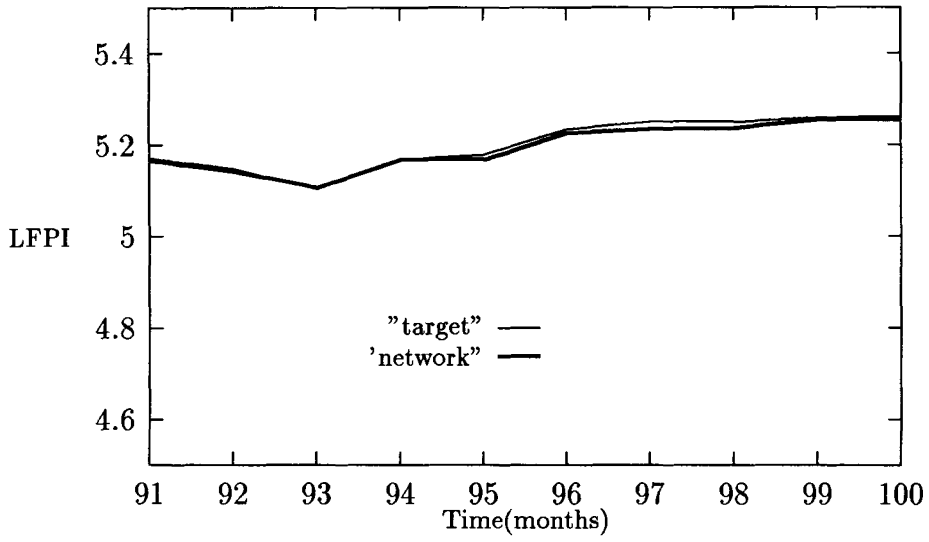

Figure 10: Network Prediction, one-lag (Buffalo)

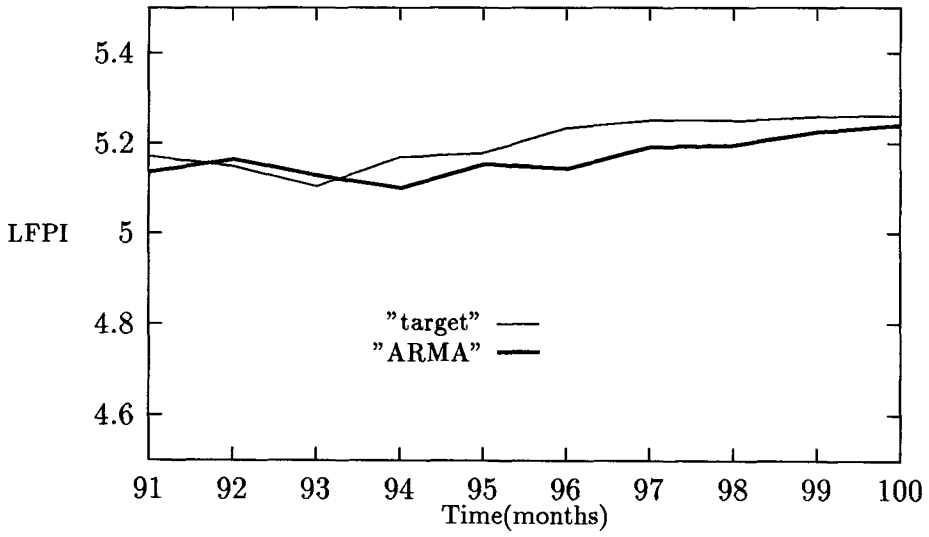

Figure 11: ARMA Prediction, one-lag (Buffalo)

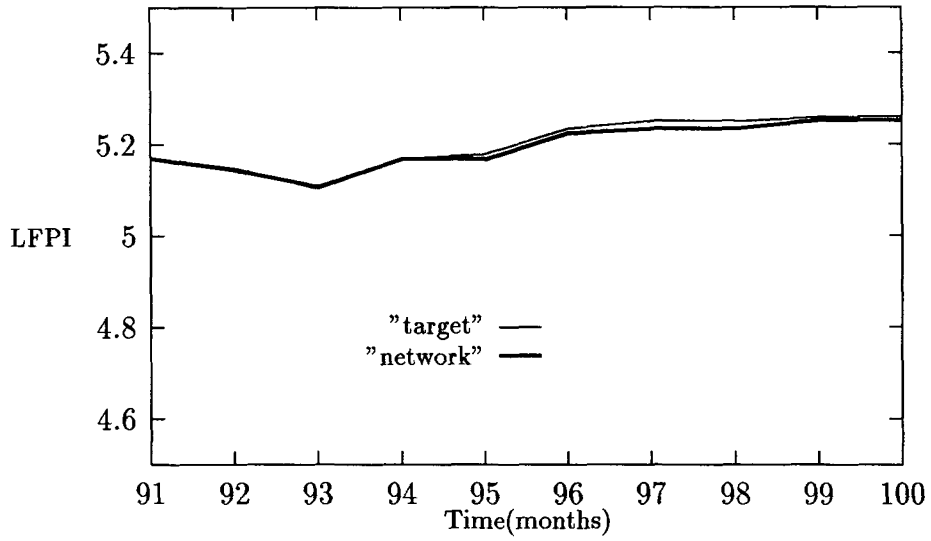

Figure 12: Network Prediction, multi-lag (Buffalo)

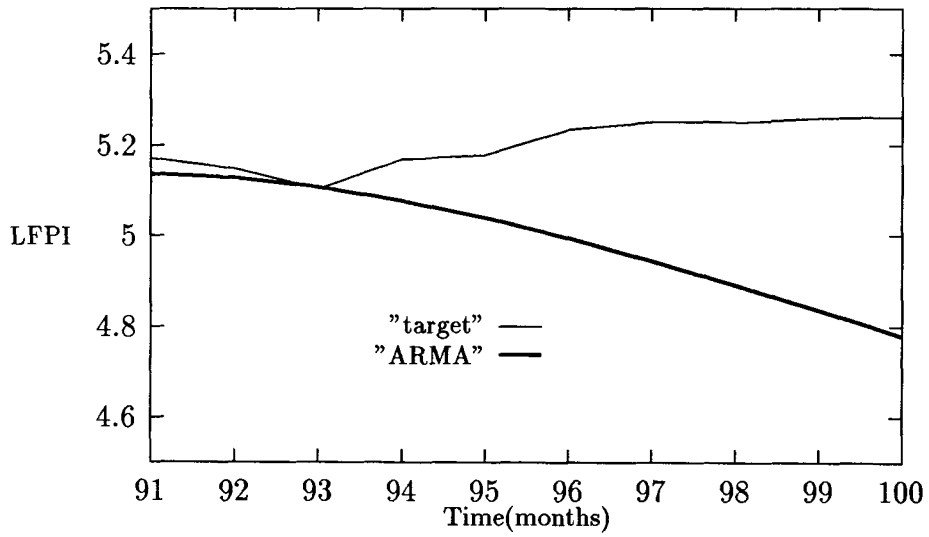

Figure 13: ARMA Prediction, multi-lag (Buffalo)

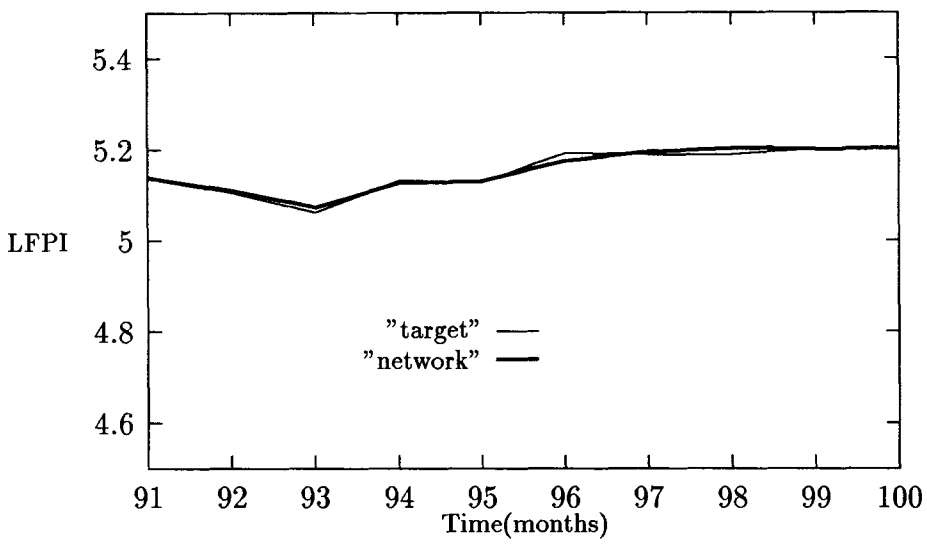

Figure 14: Network Prediction, one-lag (Minneapolis)

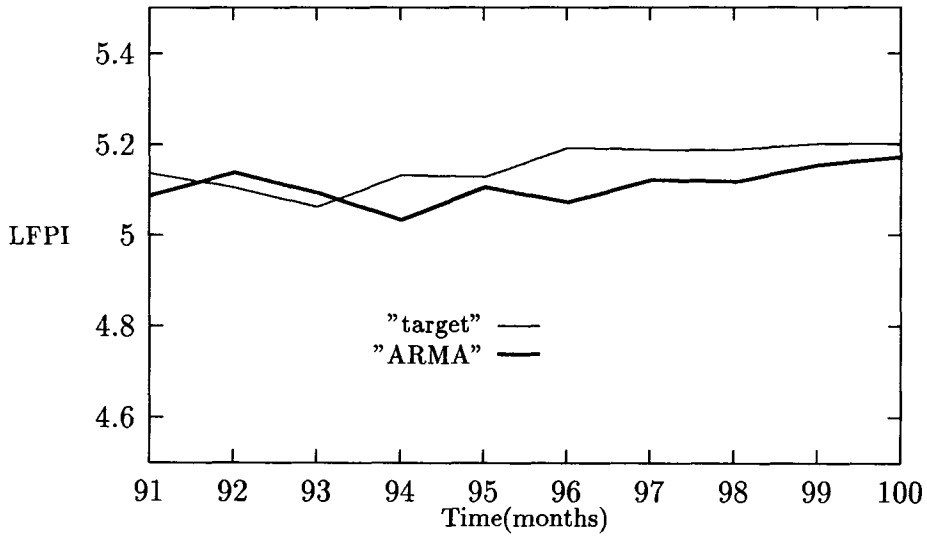

Figure 15: ARMA Prediction, one-lag (Minneapolis)

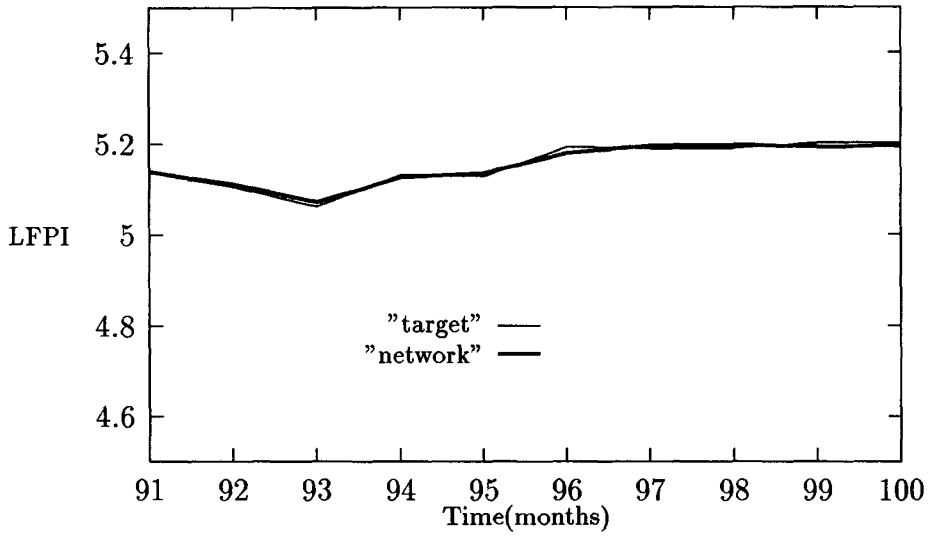

Figure 16: Network Prediction, multi-lag (Minneapolis)

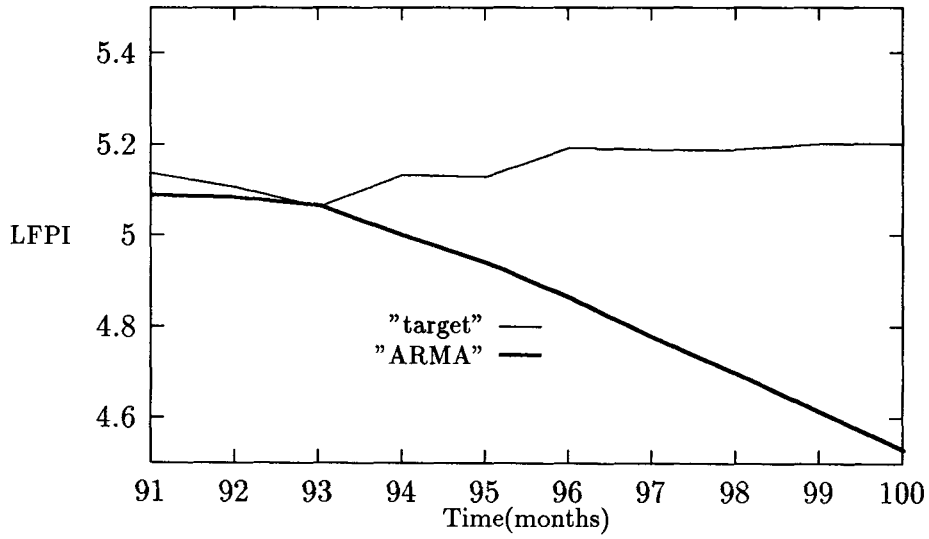

Figure 17: ARMA Prediction, multi-lag (Minneapolis)

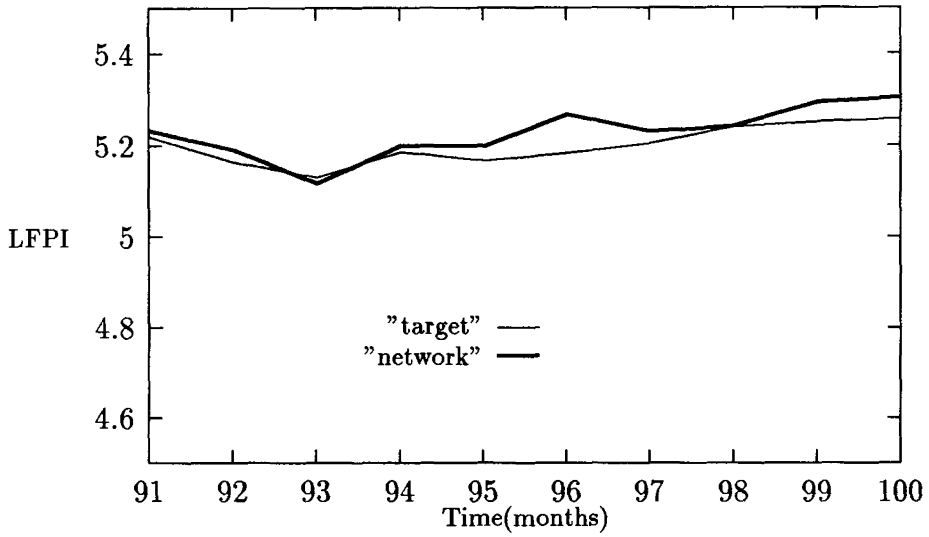

Figure 18: Network Prediction, one-lag (Kansas City)

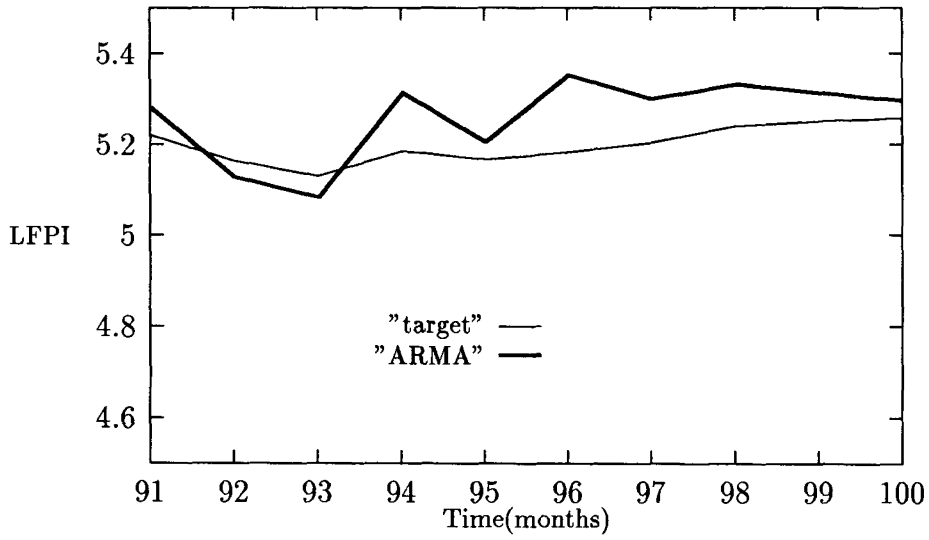

Figure 19: ARMA Prediction, one-lag (Kansas City)

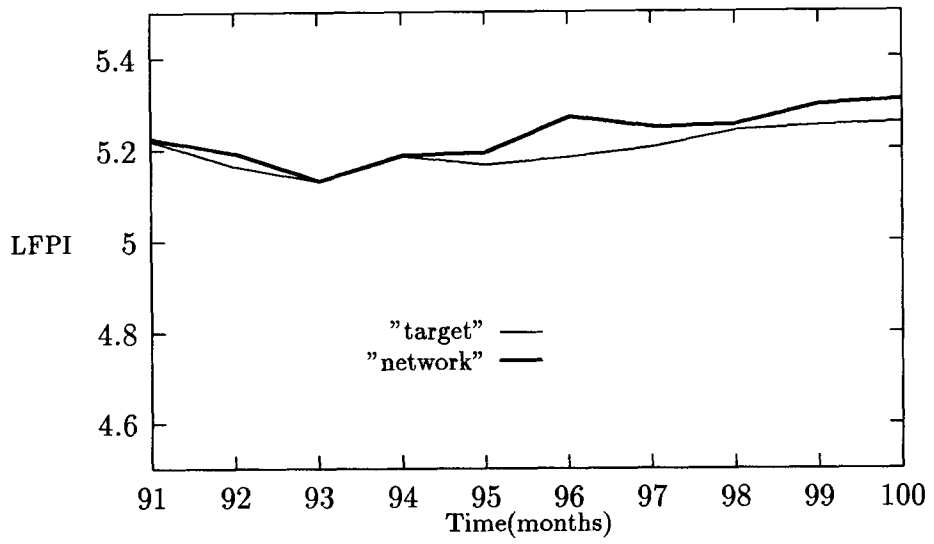

Figure 20: Network Prediction, multi-lag (Kansas City)

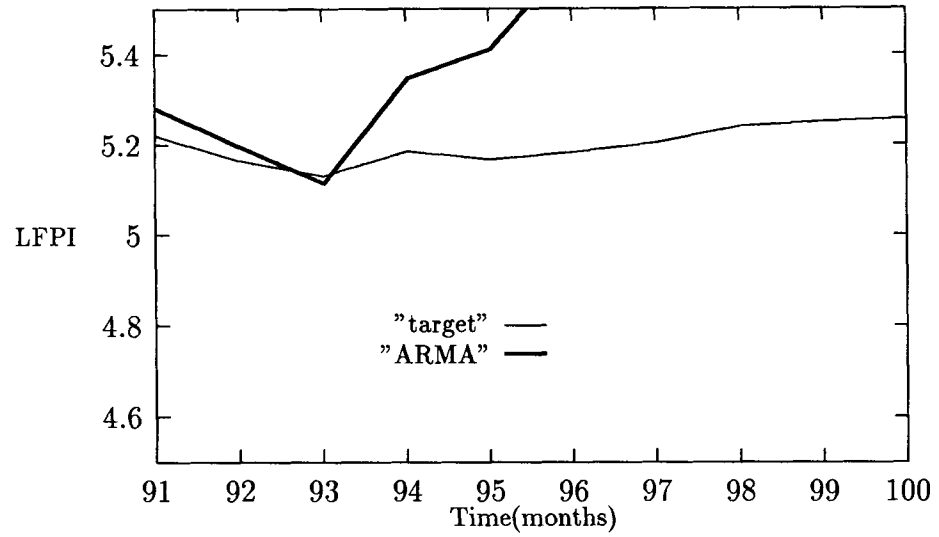

Figure 21: ARMA Prediction, multi-lag (Kansas City)

2.3 Procedure for Training the Networks

We used the error back-propagation algorithm of Rumelhart *et al.* [RuHW 86] to train the networks, with the goal of minimizing the mean squared deviation between the desired target values and network outputs, averaged over all the training inputs. In each step in the training phase, a d-tuple (recent history) of normalized flour-prices, is presented

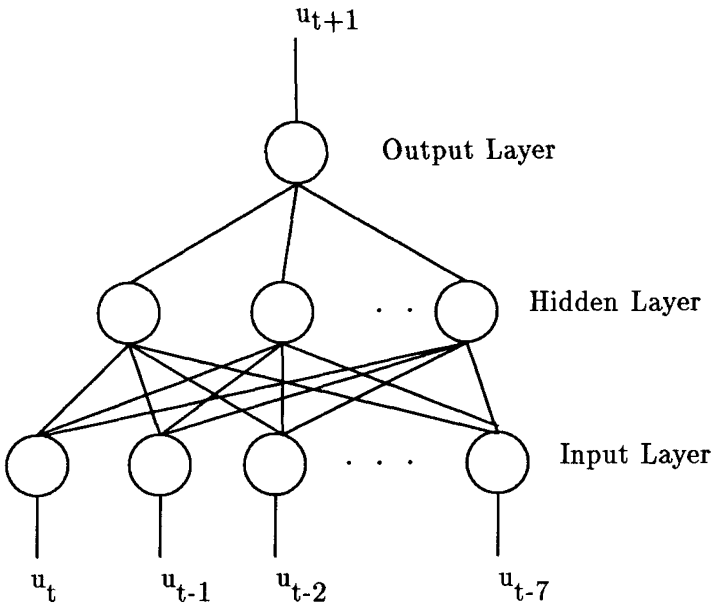

Figure 22: Single Architecture

to the network. The network is asked to predict the next value in the time sequence for the chosen city. The error between the value predicted (by the network) and the value actually observed (known data) is then measured and propagated backwards along the feedforward connections. The weights of links between units are modified to various extents, using a technique which apportions 'blame' for the error to various nodes and links, as prescribed by the back-propagation algorithm. If the mean squared error exceeds some small predetermined value, a new 'epoch' (cycle of presentations of all training inputs) is started after termination of the current epoch.

The parameters of the back-propagation algorithm are the 'learning rate' and 'momentum', which roughly describe the relative importance given to the current and past error-values in modifying connection strengths. For better performance in our experiments, we found that it was best to use a small learning rate in training the network. In all training cases we chose a learning rate of 0.3, and an associated momentum term of 0.6. The number of epochs varied between 25000 to 50000 in all cases.

3 Statistical Model

As shown in [TiTs 89], the autoregressive moving average (ARMA) model of prediction involves computation of the overall order of the temporal process with the help of normalized criterion and root tables, followed by estimation of unknown parameters. For this example, it was found that a trivariate ARMA(1,1) model or AR(2) model would be appropriate for the data. Subsequently, Tiao and Tsay [TiTs 89] obtained the model described below.

First, each trivariate input vector has to be transformed by premultiplication with a 3×3 matrix T, called the transformation matrix. The transformed data conforms to a trivariate ARMA(1,1) model of the form

$$(\mathbf{I} - \Phi_1 B)\mathbf{y_t} = \mathbf{c} + (\mathbf{I} - \Theta_1 B)\mathbf{a_t}$$

where the transformed series $\mathbf{y_t} = \mathsf{T}\mathbf{z_t}$ is a 3×1 (trivariate) column vector, B represents the usual backshift operator, and the 3×1 column vectors $\mathbf{a_t}$ comprise the error components of the model. The matrix coefficients $(\mathbf{I} - \Phi_1 B)$ and $(\mathbf{I} - \Theta_1 B)$ represent the autoregressive and moving average components respectively. The estimated values of the 3×3 matrices Φ_1, Θ_1, and the 3×1 vector $\mathbf{c}$ are given in [TiTs 89].

In the trivariate ARMA model, the mean squared errors are obtained from the trivariate $\mathbf{a_t}$'s, after premultiplying each such vector, for $t = 1, 2 \ldots$ by the inverse of the transformation matrix T. The manner of computing the $\mathbf{a_t}$ vectors is as follows. We initialize $\mathbf{a_1}$ to zero, and then, by using known values of $\mathbf{y_t}$, $\mathbf{a_t}$, compute $\mathbf{y_{t+1}}$ by the recipe of the model, for $t = 1, 2 \ldots 89$. The error vector $\mathbf{a_t}$ is obtained at each step by taking the difference between the computed and the actual values of $\mathbf{y_t}$, for $t = 2, 3 \ldots 90$. One-lag prediction is merely a continuation of the above process for $t = 91, 92 \ldots$, and multi-lag prediction is performed in a similar fashion but without considering the contribution of the $\mathbf{a_t}$ vectors for $t = 91, 92 \ldots$ to the model, because these can be computed only by using both actual and predicted data, and we are not permitted to use the former.

4 Analysis of Experimental Results

The mean squared errors for three different sets of experiments are listed in Table 1. The values correspond respectively to the mean squared errors observed for (a) the first 90 trivariate data items, which correspond to the training data for the combined modeling networks, (b) one-lag, and (c) multi-lag predictions of the combined modeling network and ARMA models. The mean squared errors for the ARMA model are generally several orders

of magnitude larger than those of the networks. Table 2 gives the respective coefficients of variation. The mean values of the data we worked with, *viz.* natural logarithms of monthly flour price indices for Buffalo, Minneapolis and Kansas City, were 5.021, 4.997 and 5.027 respectively.

The performance of the neural networks did not vary much for different choices of input sizes in the training and prediction phases of our experiments, and so the following results are fairly representative. Also, experiments showed that perturbing the choice of initial random weights of the network did not make any significant difference to the performance of the networks in learning the time series. The vastly improved performance of combined modeling over separate modeling, the results for which did not deserve mention in this paper, suggests the existence of high positive correlations between the temporal patterns for the three cities.

Table 1: Mean-Squared Errors $\times 10^3$

	Network Modeling/Prediction			ARMA Modeling/Prediction		
Cities	Training	One-lag	Multi-lag	Training	One-lag	Multi-lag
Buffalo	0.103	0.087	0.107	2.549	2.373	72.346
Minneapolis	0.090	0.072	0.070	5.097	4.168	137.534
Kansas City	0.383	1.353	1.521	8.645	7.497	233.413

Table 2: Coefficients of Variation $\times 10^3$

	Network Modeling/Prediction			ARMA Modeling/Prediction		
Cities	Training	One-lag	Multi-lag	Training	One-lag	Multi-lag
Buffalo	2.021	1.857	2.059	10.054	9.701	53.564
Minneapolis	1.898	1.697	1.674	14.285	12.917	74.204
Kansas City	3.892	7.316	7.757	18.493	17.222	96.096

5 Discussion

Most statistical models for learning and predicting time series are based only on linear recurrences. Though computationally inexpensive, such functions do not often accurately represent temporal variations. Nonlinear functions, on the other hand, are more useful for tracing temporal sequences. This is probably the main reason for the significantly better

performance of the neural network approach (with nonlinearities at each node) as compared to statistical modeling.

In any non-trivial time series, new values depend not only on the immediately prior value, but also on many preceding values. Using too few inputs can result in inadequate modeling, whereas too many inputs can excessively complicate the model. In the context of neural networks, too many inputs would imply slower training and slower convergence, and may in fact worsen the generalization capabilities (applicability to test cases) of the network. Weigend *et al.* [WAHR 90] have given a rule of thumb for determining the number of weights in the network as a function of the number of training samples. But this rule was found to be too restrictive for the data set of 100 patterns we worked with, and hence had to be disregarded.

Different types of connectionist models have been proposed for learning temporal variations of data. It has generally been held in the past that recurrent networks are more suitable for learning temporal data. There were two reasons why recurrent networks were not used for modeling the trivariate data on flour-prices – we observed experimentally that unfolding them into simple feedforward networks would cause worse training and output predictions than single hidden layer feedforward nets; the network would become inherently slower because of much greater amount of computation involved. It may be noted in passing that an unfolded version of a recurrent network is an approximation of it and implementing an exact recurrent network is a computationally expensive task. The main reason for this is that the units in hidden layers must be made to iterate among themselves till their outputs converge, and there is no way of knowing *a priori* how many iterations it would take before all the hidden units have stable outputs. Simple feedforward nets are much less computationally intensive and give good performance in less time.

A potential objection to the claim of improved performance using the neural network approach, in comparison to the statistical approach, is that neural networks are more complex and have many more parameters (weights and thresholds): would a more complex statistical model perform equally well? The answer is essentially methodological. Often, the real-world phenomena being modeled are so complex that it is impossible to theorize and generate statistical models. A large investment of experts' domain-specific research studies must precede the formulation of an adequate model for each separate phenomenon. When a large number of parameters are involved, it is difficult to predict data even when the laws which govern their behavior are known, *e.g.*, in the gravitational interactions between a large number of bodies. With neural networks, on the contrary, an essentially similar

architecture can be quickly modified and trained for a variety of different phenomena. The procedure is data-driven rather than model-driven and gives good results in many cases despite the unavailability of a good theory/model underlying the observed phenomenon.

We now evaluate the neural network approach with respect to the following criteria for a good model suggested in the literature [Harv 89]:

1. *Parsimony:* The neural network does contain a large number of parameters and is hence not parsimonious. However, the method of training does not impose any external biases, and networks started with different random weights successfully converged to approximate the time series very well.

2. *Data coherence:* The neural network model provides a very good fit with the data, as shown by the low mean-squared-error values for the training samples.

3. *Consistency with prior knowledge:* No explicit theory was constructed using the neural networks, hence this criterion is largely irrelevant. In the best neural network model, the assumption that flour prices become known in a fixed order $(x_1, y_1, z_1, x_2, \ldots)$ is consistent with the information that the data are available slightly earlier for some cities than for others.

4. *Data admissibility:* The values in a time series predicted by the neural networks are always close to the immediately preceding values, and do not violate any obvious definitional or reasonable constraints.

5. *Structural stability:* The neural networks satisfy this criterion, because they give a good fit for test data, which are outside the set of training samples.

6. *Encompassing:* The results obtained using the neural networks are better than those obtained using the alternative ARMA models. However, no theory is directly suggested by the neural networks developed so far. The design of forecasting models utilizing the parameters of the trained neural networks is currently under way.

5.1 Conclusions

We have presented a neural network approach to multivariate time-series analysis. In our experiments, real world observations of flour prices in three cities have been used to train and test the predictive power of feedforward neural networks. Remarkable success has been achieved in training the networks to learn the price curve for each of these cities, and thereby

to make accurate price predictions. Our results show that the neural network approach leads to better predictions than a well-known autoregressive moving average (ARMA) model [TiTs 89]. We obtained a very close fit during the training phase, and the networks we developed consistently outperformed statistical models during the prediction phase. Our methodology is not problem specific, and can be applied to other problems in the fields of dynamical system modeling, recognition, prediction and control.

We are currently exploring the combination of statistical and neural approaches for time-series analyses. We expect that model-based statistical preprocessing can further improve the performance or help in obtaining faster convergence of neural networks in the task of time series predictions.

References

[BoJe 70] G.E.P.Box, G.M.Jenkins, *Time Series Analysis: forecasting and control,* Holden-Day, San Francisco, 1970.

[Harv 89] A.C.Harvey, *Forecasting, Time Series Models and the Kalman Filter,* Cambridge Univ. Press, U.K., 1989.

[LaFa 87] A.S.Lapedes, R.M.Farber, *Nonlinear signal processing using neural networks: prediction and system modeling,* Technical Report LA-UR-87-2662, Los Alamos National Laboratory, 1987.

[LMMR 90] M.Li, K.Mehrotra, C.K.Mohan, S.Ranka, *Forecasting Sunspot Numbers Using Neural Networks,* Proc. IEEE Symp. on Intelligent Control, Sept. 1990.

[RuHW 86] D.E.Rumelhart, G.E.Hinton, R.J.Williams, *Learning internal representations by error propagation,* in *Parallel Distributed Processing* (eds: D.E.Rumelhart, J.L. McClelland), MIT Press, 1986.

[TiTs 89] G.C,Tiao, R.S.Tsay, *Model Specification in Multivariate Time Series,* J.R. Statist. Soc. B(1989) 51, No. 2, pp 157-213.

[WAHR 90] A.S.Weigend, B.A.Huberman, D.E.Rumelhart, *Predicting the Future: A Connectionist Approach,* submitted to the International Journal of Neural Systems, April 1990.