A New Generic Architecture For Time Series Prediction

Nageswara Rao Puli, **Nagul Shaik,M.Kishore Kumar**
*Dept. of Computer Science & Engg
Nimra Institute of Science and Technology
Vijayawada, India
**Asst.Professor, Dept. of Computer Science & Engg
Nimra Institute of Science and Technology
Vijayawada, India
***Professor & HOD Dept. of Computer Science & Engg
Nimra Institute of Science and Technology
Vijayawada, India

Abstract

Rapidly evolving businesses generate massive amounts of time-stamped data sequences and cause a demand for both univari- ate and multivariate time series forecasting. For such data, traditional predictive models based on autoregression are often not sufficient to capture complex non-linear relationships between multidimensional fea- tures and the time series outputs. In order to exploit these relationships for improved time series forecasting while also better dealing with a wider variety of prediction scenarios, a forecasting system requires a flexible and generic architecture to accommodate and tune various individual predictors as well as combination methods.

In reply to this challenge, an architecture for combined, multilevel time series prediction is proposed, which is suitable for many different universal regressors and combination methods. The key strength of this architecture is its ability to build a diversified ensemble of individual predictors that form the input to a multilevel selection and fusion process before the final optimised output is obtained. Excellent generalisation ability is achieved due to the highly boosted complementarity of indi- vidual models further enforced through crossvalidation-linked training on exclusive data subsets and ensemble output post-processing. In a sample configuration with basic neural network predictors and a mean combiner, the proposed system has been evaluated in different scenarios and showed a clear prediction performance gain.

Index Terms— Time series forecasting, combining predictors, regression, ensembles, neural networks, diversity

Introduction

Time series forecasting, or time series prediction, takes an existing series of data $x_{t-n}, \dots, x_{t-2}, x_{t-1}, x_t$ and forecasts the $x_{t+1}, x_{t+2}, \dots$ data values. The goal is to observe or model the existing data series to enable future unknown data values to be forecasted accurately. Examples of data series include financial data series (stocks, indices, rates, etc.), physically observed data series (sunspots, weather, etc.), and mathematical data series (Fibonacci sequence, integrals of differential equations, etc.). The phrase “time series” generically refers to any data series, whether or not the data are dependent on a certain time increment.
Throughout the literature, many techniques have been implemented to perform time series forecasting. This paper will focus on two techniques: *neural networks* and *k-nearest-neighbor*. This paper will attempt to fill a gap in the abundant neural network time series forecasting literature, where testing arbitrary neural networks on arbitrarily complex data series is common, but not very enlightening. This paper thoroughly analyzes the responses of specific neural network configurations to artificial data series, where each data series has a specific characteristic. A better understanding of what causes the basic neural network to become an inadequate forecasting technique will be gained. In addition, the influence of data preprocessing will be noted. The forecasting performance of k-nearest-neighbor, which is a much simpler forecasting technique, will be compared to the neural networks’ performance. Finally, both techniques will be used to forecast a real data series.

Difficulties

Several difficulties can arise when performing time series forecasting. Depending on the type of data series, a particular difficulty may or may not exist. A first difficulty is a *limited quantity of data*. With data series that are observed, limited data may be the foremost difficulty. For example, given a company’s stock that has been publicly traded for one year, a very limited amount of data are available for use by the forecasting technique.

A second difficulty is *noise*. Two types of noisy data are (1) erroneous data points and (2) components that obscure the underlying form of the data series. Two examples of erroneous data are measurement errors and a change in measurement methods or metrics. In this paper, we will not be concerned about erroneous data points. An example of a component that obscures the underlying form of the data series is an additive high-frequency component. The technique used in this paper to reduce or remove this type of noise is the moving average. The data series $\ldots, x_{t-4}, x_{t-3}, x_{t-2}, x_{t-1}, x_t$ becomes $\ldots, [(x_{t-4} + x_{t-3} + x_{t-2})/3], [(x_{t-3} + x_{t-2} + x_{t-1})/3], [(x_{t-2} + x_{t-1} + x_t)/3]$ after taking a moving average with an interval *i* of three. Taking a moving average reduces the number of data points in the series by $i-1$.

A third difficulty is *nonstationarity*, data that do not have the same statistical properties (e.g., mean and variance) at each point in time. A simple example of a nonstationary series is the Fibonacci sequence: at every step the sequence takes on a new, higher mean value. The technique used in this paper to make a series stationary in the mean is first-differencing. The data series $\ldots, x_{t-3}, x_{t-2}, x_{t-1}, x_t$ becomes $\ldots, (x_{t-2} - x_{t-3}), (x_{t-1} - x_{t-2}), (x_t - x_{t-1})$ after taking the first-difference. This usually makes a data series stationary in the mean. If not, the second-difference of the series can be taken. Taking the first-difference reduces the number of data points in the series by one.

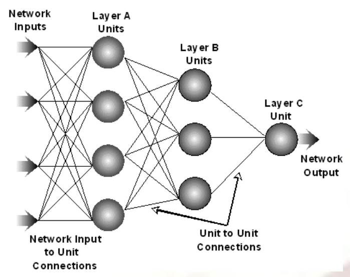
A fourth difficulty is *forecasting technique selection*. From statistics to artificial intelligence, there are myriad choices of techniques. One of the simplest techniques is to search a data series for similar past events and use the matches to make a forecast. One of the most complex techniques is to train a model on the series and use the model to make a forecast. K-nearest-neighbor and neural networks are examples of the first and second techniques, respectively.

1) Importance

Time series forecasting has several important applications. One application is preventing undesirable events by forecasting the event, identifying the circumstances preceding the event, and taking corrective action so the event can be avoided. At the time of this writing, the Federal Reserve Committee is actively raising interest rates to head off a possible inflationary economic period. The Committee possibly uses time series forecasting with many data series to forecast the inflationary period and then acts to alter the future values of the data series.Another application is forecasting undesirable, yet unavoidable, events to preemptively lessen their impact. At the time of this writing, the sun’s cycle of storms, called solar maximum, is of concern because the storms cause technological disruptions on Earth. The sunspots data series, which is data counting dark patches on the sun and is related to the solar storms, shows an eleven-year cycle of solar maximum activity, and if accurately modeled, can forecast the severity of future activity. While solar activity is unavoidable, its impact can be lessened with appropriate forecasting and proactive action.

Finally, many people, primarily in the financial markets, would like to profit from time series forecasting. Whether this is viable is most likely a never-to-be-resolved question. Nevertheless many products are available for financial forecasting.
Difficulties inherent in time series forecasting and the importance of time series forecasting are presented next. Then, neural networks and k-nearest-neighbor are detailed. Section **Error! Reference source not found.** presents related work. Section **Error! Reference source not found.** gives an application level description of the test-bed application, and Section **Error! Reference source not found.** presents an empirical evaluation of the results obtained with the application.A time series is a sequence of observations of a random variable. Hence, it is a stochasticprocess. Examples include the monthly demand for a product, the annual freshmanenrollment in a department of a university, and the daily volume of flows in a river.Forecasting time series data is important component of operations research because thesedata often provide the foundation for decision models. An inventory model requiresestimates of future demands, a course scheduling and staffing model for a universityrequires estimates of future student inflow, and a model for providing warnings to thepopulation in a river basin requires estimates of river flows for the immediate future.Time series analysis provides tools for selecting a model that can be used to forecastof future events. Modeling the time series is a statistical problem. Forecasts are used incomputational procedures to estimate the parameters of a model being used to allocatedlimited resources or to describe random processes such as those mentioned above. Timeseries models assume that observations vary according to some probability distributionabout an underlying function of time.styles are built-in; examples of the type styles are provided throughout this document and are identified in italic type, within parentheses, following the example. PLEASE DO NOT RE-ADJUST THESE MARGINS.

In this section and the next, subscripts *c*, *p*, and *n* will identify units in the current layer, the previous layer, and the next layer, respectively. When the network is run, each hidden layer unit performs the calculation in **Error! Reference source not found.** on its inputs and transfers the result (O_c) to the next layer of units.

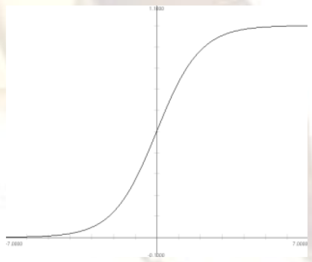


$$O_c = h_{Hidden}\left(\sum_{p=1}^{P} i_{c,p} w_{c,p} + b_c\right) \text{ where } h_{Hidden}(x) = \frac{1}{1+e^{-x}}$$

Fig: Forwarding the node values

O_c is the output of the current hidden layer unit *c*, *P* is

either the number of units in the previous hidden layer or number of network inputs, $i_{c,p}$ is an input to unit *c* from either the previous hidden layer unit *p* or network input *p*, $w_{c,p}$ is the weight modifying the connection from either unit *p* to unit *c* or from input *p* to unit *c*, and b_c is the bias.
In **Error! Reference source not found.**, $h_{Hidden}(x)$ is



the sigmoid *activation function* of the unit and is charted in **Error! Reference source not found.**.

Fig: Prediction Graph for Forwarding node

Other types of activation functions exist, but the sigmoid was implemented for this research. To avoid saturating the activation function, which makes training the network difficult, the training data must be scaled appropriately. Similarly, before training, the weights and biases are initialized to appropriately scaled values.

. Each output layer unit performs the calculation in Equation II.**1** on its inputs and transfers the result (O_c) to a network output.

Equation II.1 *Activation function of an output layer unit.*
Each output layer unit performs the calculation in Equation II.**1** on its inputs and transfers the result (O_c) to a network output.

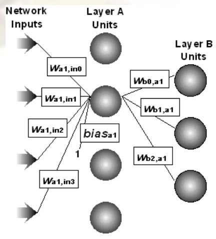
Equation II.2 *Activation function of an output layer unit.*

$$O_c = h_{Output}\left(\sum_{p=1}^{P} i_{c,p} w_{c,p} + b_c\right) \text{ where } h_{Output}(x) = x$$

Oc is the output of the current output layer unit c, P is the number of units in the previous hidden layer, ic,p is an input to unit c from the previous hidden layer unit p, wc,p is the weight modifying the connection from unit p to unit c, and bc is the bias. For this research, hOutput(x) is a linear activation function1

K-Nearest-Neighbor

In contrast to the complexity of the neural network forecasting technique, the simpler k-nearest-neighbor forecasting technique is also implemented and tested. K-nearest-neighbor is simpler because there is no model to train on the data series. Instead, the data series is searched for situations similar to the current one each time a forecast needs to be made.
To make the k-nearest-neighbor process description easier, several terms will be defined. The final data points of the data series are the reference, and the length of the reference is the window size. The data series without the last data point is the shortened data series. To forecast the data series' next data point, the reference is compared to the first group of data points in the shortened data series, called a candidate, and an error is computed. Then the reference is moved one data point forward to the next candidate and another error is computed, and so on. All errors are stored and sorted. The smallest k errors correspond to the k candidates that closest match the
Fig: Nearest Neighbor Transformation



reference. Finally, the forecast will be the average of the k data points that follow these candidates. Then, to forecast the next data point, the process is repeated with the previously forecasted data point appended to the end of the data series.

II. CONCLUSION

Section **Error! Reference source not found.** introduced time series forecasting, described the work presented in the typical neural network paper, which justified this paper, and identified several difficulties associated with time series forecasting. Among these difficulties, noisy and nonstationary data were investigated further in this paper. Section **Error! Reference source not found.** also presented feed-forward neural networks and backpropagation training, which was used as the primary time series forecasting technique in this paper. Finally, k-nearest-neighbor was presented as an alternative forecasting technique.

Section **Error! Reference source not found.** briefly discussed previous time series forecasting papers. The most notable of these being the paper by Drossu and Obradovic (1996), who presented compelling research combining stochastic techniques and neural networks. Also of interest were the paper by Geva (1998) and the book by Kingdon (1997), which took significantly more sophisticated approaches to time series forecasting.

Section **Error! Reference source not found.** presented Forecaster and went through several important aspects of its design, including parsing data files, using the Wizard to create networks, training networks, and forecasting using neural networks and k-nearest-neighbor.

Section **Error! Reference source not found.** presented the crux of the paper. First, the data series used in the evaluation were described, and then parameters and procedures used in forecasting were given. Among these was a method for selecting the number of neural network inputs based on data series characteristics (also applicable to selecting the window size for k-nearest-neighbor), a training heuristic, and a metric for making quantitative forecast comparisons. Finally, a variety of charts and tables, accompanied by many empirical observations, were presented for networks trained heuristically and simply and for k-nearest-neighbor.

REFERENCES

[1]. Drossu, R., & Obradovic, Z. (1996). Rapid Design of Neural Networks for Time Series Prediction. IEEE Computational Science & Engineering, Summer 1996, 78-89.

[2]. Geva, A. (1998). ScaleNet—Multiscale Neural-Network Architecture for Time Series Prediction. IEEE Transactions on Neural Networks, 9(5), 1471-1482.

[3]. Gonzalez, R. C. & Woods, R. E. (1993). Digital Image Processing. New York: Addison-Wesley.

[4]. Hebb, D. O. (1949). The Organization of Behavior: A Neuropsychological Theory. New York: Wiley & Sons.

[5]. Kingdon, J. (1997). Intelligent Systems and Financial Forecasting. New York: Springer-Verlag.

[6]. Lawrence, S., Tsoi, A. C., & Giles, C. L. (1996). Noisy Time Series Prediction Using Symbolic Representation and Recurrent Neural Network Grammatical Inference [Online]. Available: http://www.neci.nj.nec.com/homepages/lawrence/papers/finance-tr96/latex.html [March 27, 2000].

[7]. McCulloch, W. S., & Pitts, W. H. (1943). A Logical Calculus of the Ideas Imminent in Nervous Activity. Bulletin of Mathematical Biophysics, 5, 115-133.

[8]. Minsky, M., & Papert, S. (1969). Perceptrons: An Introduction to Computational Geometry. Cambridge, MA: MIT Press.

[9]. Rosenblatt, F. (1962). Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Washington, D. C.: Spartan.

[10]. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning Internal Representations by Error Propagation. In D. E. Rumelhart, et al. (Eds.), Parallel Distributed Processing: Explorations in the Microstructures of Cognition, 1: Foundations, 318-362. Cambridge, MA: MIT Press.

[11]. Torrence, C., & Compo, G. P. (1998). A Practical Guide to Wavelet Analysis [Online]. Bulletin of the American Meteorological Society. Available: http://paos.colorado.edu/research/wavelets/ [July 2, 2000].

[12]. Zhang, X., & Thearling, K. (1994). Non-Linear Time-Series Prediction by Systematic Data Exploration on a Massively Parallel Computer [Online]. Available: http://www3.shore.net/~kht/text/sfitr/sfitr.htm [March 27, 2000].