Neuro-Fuzzy Inference System (ASuPFuNIS) Model for Intervention Time Series Prediction of Electricity Prices

Apurva Narayan*, Keith. W. Hipel*, K. Ponnambalam* and Sandeep Paul†
*Department of Systems Design Engineering
University of Waterloo,
Waterloo, ON Canada N2L 2G1
Email:apurva.narayan@uwaterloo.ca
†Dayalbagh Educational Institute, Dayalbagh, Agra - 282110 INDIA

***Abstract*—This paper presents an approach to time series prediction based on Asymmetric Subsethood-Product Fuzzy Neural Inference System (ASuPFuNIS). The standard time series techniques have standard averaging where a fixed weight is added to the past values. In this paper we present a novel neuro-fuzzy inference system based on asymmetric subsethood with intervention based transfer function based time series model for accurate prediction of time series. The design of the model is described, and the scheme is evaluated by application to real-world problem of cost of electricity prices over a period of seven year in Ontario, Canada. We also study the various statistical properties of the data.**

I. Introduction

Time series is a very popular method for the analysis of sequentially organized observations so that the governing characteristics of the process are uncovered. Time series is the data which is observed at regular intervals, and it is generated in various fields concerned with our daily life such as weather, stock market, and electric power, etc. The essential objective of time series analysis is to forecast the future behaviour of a system based on the historical information. This analysis results in the description of the process through a number of equations that in principle combine the current values of the series, y_t, with past observations, y_{t-k}, exogenous variables, x_{t-j}, and previous error terms, e_{t-k}. The generalized form of this process becomes as explained in [1]

$$y_t = f(y_{t-k}, x_{t-j}, e_{t-k}) \quad (1)$$

As clearly mentioned in [1], a variety of statistical models such as exponential smoothing, generalized regression and ARMA have been used previously to describe the unknown function f. Another approach of fitting non-linear models that are widely available such as artificial intelligence models and chaotic processes is used. Despite the availability of strong forecasting methods already available there is a thrust towards the development of better forecasting schemes. One major drawback in current techniques is the amount of information needed to explain the variation of the series.

Analysis of time series is needed before the prediction can be done. Time series analysis is to find out the various probabilistic and statistical characteristics in them. Linear regression models (such as Auto Regressive (AR), Moving Average (MA) and Auto Regressive Moving Average (ARMA)) and Box-Jenkins Model have been most widely used for time series analysis[2][3]. Recently, during the past few years researchers have been exploring the soft computing techniques such as fuzzy theory, neural networks and genetic algorithms [4].

A novel technique based on the least squares support vector machines (LS-SVM) [5] was proposed to make one-step or multi-step prediction of chaotic time series. Also various filtering methods have been employed in the prediction of chaotic time series such as neural Volterra filter [6] and Kalman filter [7].

Numerous examples of synergistic fuzzy neural models that combine the merits of connectionist and fuzzy approaches have been proposed in the literature [8], [9], [10]. These include the fuzzy multilayer perceptron [11]; neural fuzzy systems [12]; and evolvable neuro-fuzzy systems [13]. By virtue of their ability to refine initial domain knowledge, operate and adapt in both numeric as well as linguistic environments, these models are extensively used in a wide range of application domains such as approximate reasoning and inferencing [14]; classification [15]; control [16]; rule extraction and simplification [17].

Most hybrid models embed data-driven or expert derived knowledge, in the form of fuzzy *if-then* rules, into a network architecture to facilitate fast learning [18]. This embedding of knowledge is often done by assuming that antecedent and consequent labels of standard fuzzy *if-then* rules are represented as connection weights of the network as in [19]. The composition of the incoming numeric information with this embedded knowledge is usually done by computing membership values from fuzzy membership functions that represent network weights [15] or by fuzzifying the numeric inputs by modeling them as fuzzy numbers using triangular or Gaussian membership functions and then using the well defined sup-star or mutual subsethood composition mechanisms [18], [20]. In case of symbolic information, a given universe of discourse is generally quantized into pre-specified fuzzy sets. A fuzzy input is then simply one of these pre-specified fuzzy sets [11], [21]. Subsequently a learning algorithm fine tunes these rules based on the available training data that describes the problem. Commonly used learning algorithms employ either supervised gradient descent and their variants [11], unsupervised learning, reinforcement learning, or genetic algorithm based search [22], [16].

Asymmetric Subsethood-Product Fuzzy Neural Inference System (ASuPFuNIS) directly extends SuPFuNIS [18] by permitting the signal and weight fuzzy sets to be asymmetric in the sense that the left and right spreads of the signal and weight fuzzy sets can differ. Both signal and weight fuzzy sets are thus defined by three parameters: a center, a left spread and a right spread. This extension is motivated by the fact that asymmetric fuzzy sets lend flexibility and thereby may be fruitful in capturing the non-uniformity of data in complex problems.

The rest of the paper is organized as Section II talks about Methodology , Section III gives the background information of the data and its statistical properties, Section IV shows the implementation strategy, Section V shows some results, Section VI concludes the paper and expresses ideas on future development.

II. Methodology

The ASuPFuNIS network is trained by supervised learning, once again in a way similar to that of SuPFuNIS [18]. This involves repeated presentation of input patterns drawn from the training set and comparing the output of the network with the desired value to obtain the error. Network weights are changed on the basis of an error minimizing criterion. Once the network is trained to the desired level of error, it is tested by presenting unseen test set patterns.

Learning is incorporated into ASuPFuNIS using the standard iterative pattern based gradient descent method. The instantaneous squared error $e(t)$ at iteration t is used as a training performance parameter and is computed in the standard way:

$$e(t) = \frac{1}{2}\sum_{k=1}^{p}\left(d_k(t) - S(y_k(t))\right)^2 \qquad (2)$$

where $d_k(t)$ is the desired value at output node k, and $e(t)$ is evaluated over all p outputs for a specific input pattern $X(t)$. Notice that for an n-q-p architecture of ASuPFuNIS, the number of connections is $(nq+qp)$. Since in the proposed model, the representation of a fuzzy weight requires three parameters (center, left and right spreads) and an input feature requires two parameters (left and right spreads), the total number of free parameters to be trained will be $3(nq+qp)+2n$. If the trainable parameters of ASuPFuNIS be represented as a vector $P = \left(x_i^{\sigma^l}, x_i^{\sigma^r}, w_{ij}^c, w_{ij}^{\sigma^l}, w_{ij}^{\sigma^r}, v_{jk}^c, v_{jk}^{\sigma^l}, v_{jk}^{\sigma^r}\right)^T$, then the iterative gradient descent update equation can be written as

$$P(t+1) = P(t) - \eta\nabla e(t) + \alpha\Delta P(t-1) \qquad (3)$$

where η is the learning rate, $\nabla e(t) = \left(\frac{\partial e(t)}{\partial x_i^{\sigma^l}}, \frac{\partial e(t)}{\partial x_i^{\sigma^r}}, \frac{\partial e(t)}{\partial w_{ij}^c}, \frac{\partial e(t)}{\partial w_{ij}^{\sigma^l}}, \frac{\partial e(t)}{\partial w_{ij}^{\sigma^r}}, \frac{\partial e(t)}{\partial v_{jk}^c}, \frac{\partial e(t)}{\partial v_{jk}^{\sigma^l}}, \frac{\partial e(t)}{\partial v_{jk}^{\sigma^r}}\right)^T$ is the gradient vector, α is the momentum parameter and $\Delta P(t-1) = P(t) - P(t-1)$.

III. Background Information

The basic time series which we are going to model represents electric prices in Ontario, Canada. The electric prices tend to vary hourly at various times of the day. They are usually low during the off peak hours and are comparatively high at peak hours of the day since the electricity consumption is high during this time. These price fluctuations are dependant on various factors as cost of generation, transmission cost, regulatory costs etc.

Fluctuations in these electric prices are regulated not only by electric companies but also consumer and government agencies. In the new evolving power systems where *smart grid* is playing an important role in the electricity prices where consumer comes into the scenario in regulating electricity prices by providing electricity to the grid.

First step in time series modelling is *exploratory data analysis* where we are going to look at various plots of the data and predict a model which may most appropriately fit our time series.

A. Exploratory and Confirmatory Data Analysis

The figure 1 below indicates the general statistical behaviour of the the raw data of the time series having monthly data for seven year of average electricity prices. From the figure 1 we can infer that the electric prices have generally been stationary when averaged daily over a month. Although we can see in the year 2004 there seems to be a shoot up of prices which may be due to various reasons. Such as the cost of resources (raw material increasing) which was controlled by government by may be subsidizing the rates which eventually brought down the rates.

Now we can also see a very interesting variation in the electricity prices in around 74^{th} month where we can see a dip in the electric prices. From the background information about the scenario in the electric market in that year it seems that some new renewable sources of power and concepts of smart grids or some government regulation for subsidizing the electric prices could have been incorporated which led the power manufacturer to regulate the prices of electricity.

The figure 2 and 3 show the *Autocorrelation and Partial Autocorrelation Functions* of the data series which are helpful in deciding the MA and AR parameters to be fit to the model.

B. Interventions

This is an important feature for our data, some times there are unknown and known interventions which impact the time series. In our data in figure 1 we can see that there are most probably 2 interventions where we can see a change in the mean level which again comes back to normal. As the first intervention in figure 1 can be seen in around lag 24 and the second intervention can be seen around lag 74.

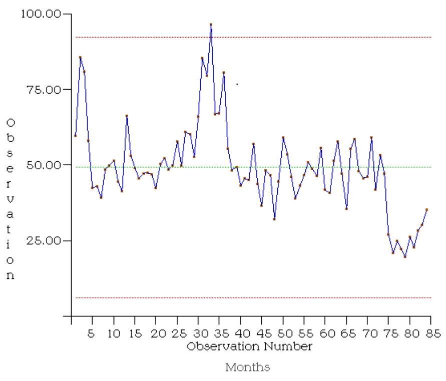


Fig. 1. This plot is of raw data

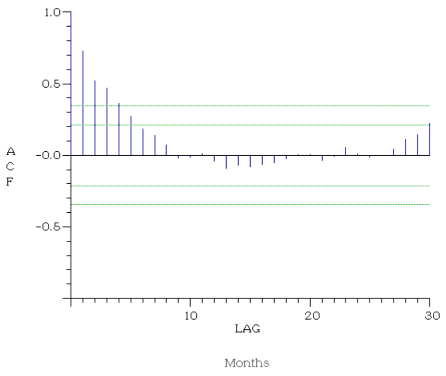


Fig. 2. Autocorrelation Function

In our case we use the Maximum Likelihood Estimator to estimate our model parameters. A detailed explanation of the MLE is given in [23] and also in [24].

The variance of the MLE of a model parameter has minimum asymptotic variance and is asymptotically normally distributed when consistency and other conditions are satisfied. There are various kinds of MLE avaible, Approximate MLEs and Exact MLEs and then we also have algorithms which are used for the optmization of the likelihood function. They have been described in detail in [24].

C. Intervention Analysis

As per the exploratory data analysis we came to a conclusion that our data underwent 2 major interventions one at around lag 24 and the other one at lag 76. On figuring out the reason behind such interventions, interesting results revealed.

Although a lot of seasonality of the data was loss due to averaging out the hourly data to monthly data over seven years. Hence from the above analysis and following the three stages of model construction from identification, parameter estimation and diagnostic checking, we find that ARMA(2,1) fitted to a 2 intervention based TFN(Transfer Function Noise) model with 1 parameter in the numerator and 1 parameter in the denominator for both the interventions is used.

The general form of 2 intervention model would be like

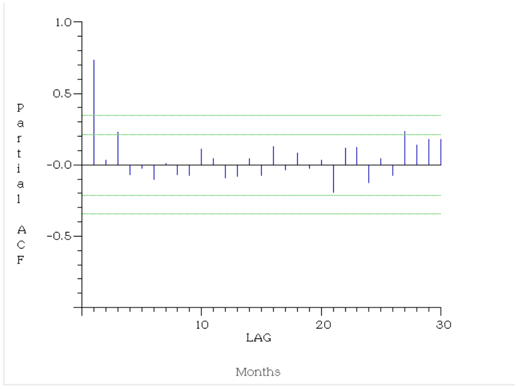


Fig. 3. Partial Autocorrelation Function

Model	AIC	Model Characteristics
Intervention TFN Model	387.7	ARIMA(2,0,1), Intervention 1 : u=2,v=2 at lag 24
		Intervention 2 : u=2,v=2 at lag 74
Intervention TFN Model	392.78	ARIMA(2,0,1), Intervention 1 : u=2,v=1 at lag 24
		Intervention 2 : u=2,v=1 at lag 74
Intervention TFN Model	388.23	ARIMA(2,0,2), Intervention 1 : u=2,v=2 at lag 24
		Intervention 2 : u=2,v=2 at lag 74
Intervention TFN Model	**386.236**	**ARIMA(2,0,1), Intervention 1 : u=1,v=1 at lag 24**
		Intervention 2 : u=1,v=1 at lag 74
Intervention TFN Model	387.23	ARIMA(3,0,1), Intervention 1 : u=1,v=1 at lag 24
		Intervention 2 : u=1,v=1 at lag 74

TABLE I
VARIOUS MODELS FITTED TO THE DATA AND THEIR AICS

this :

$$(y_t - \mu_y) = \sum_{t=1}^{2} V_i(B) + N_t$$

where N_t is the Noise term with a ARMA model

$$V_i(B) = \frac{\omega(B)}{\delta(B)} = \frac{\omega_{0i} - \omega_{1i}}{(1 - \delta_{1i} B)}$$

The parameters for the above mentioned model have been identified using the *McLeod Hipel Time Series Decision Support System* [24].

Now once we have estimated the parameters of our model using MLEs we need to discriminate between the models to choose a most accurate fit. We used various criterion to discriminate between the models . Most commonly used is *Akaike Information Criteria (AIC)*

$$AIC = -2lnML + 2k$$

where the ML = Maximum Likelihood, lnML = value of the maximized log likelihood function for a fitted model, k = number of parameters.

AIC reflects two major components :

- Model Parsimony due to 2k terms
- Good Statistical fit due to -2lnML term

We selected the model with the least AIC and this is referred to as Minimum AIC or MAIC.

Once a time series model is fit to the data we integrate the model outputs to ASuPFuNIS, a neuro-fuzzy inference system as mentioned in [25] for accurate predictions.

IV. Implementation

An indepth explanation of the algorithm is given in [25]. In [25] we can see that the ASuPFuNIS model has been used to make forecasts l time lags ahead. Here in this paper we will compare the result of the forecasting of a time series model fit using a statistical model and make MMSE (Minimum Mean Square Error) forecasts and compare it with the one step ahead forecast made by the neuro-fuzzy model ASuPFuNIS.

The electric price monthly data for a period of seven year had 84 values in the time series. From the point of view of the neuro-fuzzy model the data was divided into two halves of 42 values each. Each of them was referred to as training and test data sets. Now initially the training data was presented to the network for 10000 epochs. Once the training phase was completed the the test data was presented to the network and based on the test data the network was made to forecast the one step ahead forecast.

An interesting thing about using the one step ahead forecast is that they conserve the most information. One step ahead forecast are the one which have least amount of uncertainty and posses most information. If we can make the one step ahead forecast accurately then further forecasts can be made in the same way.

As mentioned in the table II we ran initially two sets of experiments with the above mentioned set of parameters for 10000 epochs. It was observed that the performance of the network was better when the learning rate was high with a high momentum. This is a particular case, one can iteratively run using various combinations and variations of these learning rates and momentums to get more results for analysis.

One more variation which one could experiment with such a set up is by changing the number of rules to operate. For this problem we consider a set of 5 rules to operate, for a higher precision one can always increase the number of rules and perform experiment with variations in learning rate and momentum. Various combinations of these parameter and how they can be implemented can be easily seen in in [25].

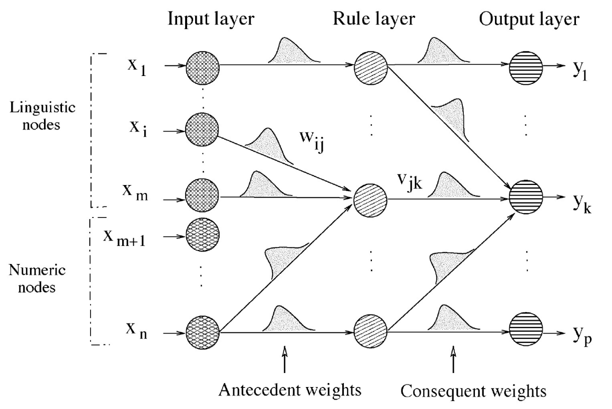
Particulary for this prediction problem, ASuPFuNIS employed a *4-q-1* network architecture : the input layer comprises of four numeric nodes; the output layer comprises a single output node and there are q rule nodes in the hidden layer. Thus, the number of trainable parameter will be $15q + 8$.

V. Architecture and Operational Details

As shown in Fig. V, ASuPFuNIS directly embeds fuzzy rules of the form

$$\text{If } x_1 \text{ is } \mathrm{A}_1^m \text{ and } x_2 \text{ is } \mathrm{A}_2^m \cdots \text{ and } x_n \text{ is } \mathrm{A}_n^m \text{ then } y_j \text{ is } \mathrm{B}_j^m \quad m = 1 \cdots q\,;\; j = 1 \cdots p \tag{4}$$

where n, q, p respectively denote the number of inputs, rules and outputs. A_i^m, B_j^m respectively represent asymmetric Gaussian fuzzy sets defined on input and output universes of discourse (UODs) for the mth rule. Each input node in the model represents a domain variable or a feature. Each hidden node represents a fuzzy rule, and input-hidden node connections represent fuzzy rule antecedents. Each output node represents a target variable or a class and hidden-output



node connections represent fuzzy rule consequents. A fuzzy weight from an input node i to a rule node j is modeled by a center w_{ij}^c, left spread $w_{ij}^{\sigma^l}$ and right spread $w_{ij}^{\sigma^r}$ of an asymmetric Gaussian fuzzy set. Thus, an antecedent connection in ASuPFuNIS is identified by a 3-tuple $w_{ij} = (w_{ij}^c, w_{ij}^{\sigma^l}, w_{ij}^{\sigma^r})$. In a similar fashion, a consequent fuzzy connection from a rule node j to an output node k is identified by a 3-tuple $v_{jk} = (v_{jk}^c, v_{jk}^{\sigma^l}, v_{jk}^{\sigma^r})$. Operationally, ASuPFuNIS is similar to SuPFuNIS [18]. In the sections that follow we present computational expressions of the model that are different. For details, the reader is referred to [18].

A. Signal Transmission at Input Nodes

Since the input feature vector $X = (x_1, \ldots, x_n)$ can comprise either numeric or linguistic values, there are two kinds of nodes in the input layer as shown in Fig. V. Linguistic nodes accept linguistic inputs represented by pre-specified asymmetric Gaussian fuzzy sets and are modeled by a center x_i^c, left spread $x_i^{\sigma^l}$ and right spread $x_i^{\sigma^r}$: $x_i = (x_i^c, x_i^{\sigma^l}, x_i^{\sigma^r})$. Often since a human expert's understanding of fuzzy linguistic terms is too vague to decide appropriate values of $x_i^{\sigma^l}, x_i^{\sigma^r}$, ASuPFuNIS tunes these membership function parameters during learning. The signal $S(x_i) = x_i$ is transmitted out of linguistic node i without transformation in the input layer.

Numeric nodes are tunable feature-specific fuzzifiers. A numeric input is fuzzified at a numeric node by treating it as the center x_i^c of an asymmetric Gaussian membership function (this is similar to the idea employed in [26], [20] where the incoming numeric inputs are fuzzified using either symmetric Gaussian or triangular fuzzy sets) with tunable left and right spreads $x_i^{\sigma^l}$ and $x_i^{\sigma^r}$. Therefore, the signal transmitted from a numeric node of the input layer is also represented by $S(x_i) = x_i = (x_i^c, x_i^{\sigma^l}, x_i^{\sigma^r})$. These fuzzy signals from the input layer are transmitted to hidden rule nodes through fuzzy weights $w_{ij} = (w_{ij}^c, w_{ij}^{\sigma^l}, w_{ij}^{\sigma^r})$ that correspond to rule antecedents.

B. Mutual Subsethood

As was done in SuPFuNIS [18] the net value of the signal transmitted along the antecedent weight is computed using a mutual subsethood measure. Consider two fuzzy sets A and

B described by asymmetric Gaussian membership functions with centers c_1, c_2, left spreads σ_1^l, σ_2^l and right spreads σ_1^r, σ_2^r respectively:

$$a(x) = \begin{cases} e^{-((x-c_1)/\sigma_1^l)^2} & -\infty < x \leq c_1 \\ e^{-((x-c_1)/\sigma_1^r)^2} & c_1 \leq x < \infty \end{cases} \tag{5}$$

$$b(x) = \begin{cases} e^{-((x-c_2)/\sigma_2^l)^2} & -\infty < x \leq c_2 \\ e^{-((x-c_2)/\sigma_2^r)^2} & c_2 \leq x < \infty. \end{cases} \tag{6}$$

The mutual subsethood $\mathcal{E}(A, B)$ [27], measures the degree to which fuzzy set A equals fuzzy set B, and is quantified as

$$\mathcal{E}(A, B) = \frac{\mathcal{C}(A \cap B)}{\mathcal{C}(A) + \mathcal{C}(B) - \mathcal{C}(A \cap B)} \in [0,\, 1], \tag{7}$$

where $\mathcal{C}(\cdot)$ denotes the cardinality of a fuzzy set which is defined for fuzzy set A by

$$\mathcal{C}(A) = \int_{-\infty}^{c_1} e^{-((x-c_1)/\sigma_1^l)^2} dx + \int_{c_1}^{\infty} e^{-((x-c_1)/\sigma_1^r)^2} dx. \tag{8}$$

Depending upon the relative values of centers and spreads of fuzzy sets A and B, six different cases of overlap can arise.

- **Case 1A**: $c_1 = c_2,\ \sigma_1^l \leq \sigma_2^l,\ \sigma_1^r \leq \sigma_2^r$; **Case 1B**: $c_1 = c_2,\ \sigma_1^l \geq \sigma_2^l,\ \sigma_1^r \geq \sigma_2^r$.
- **Case 2A**: $c_1 = c_2,\ \sigma_1^l < \sigma_2^l,\ \sigma_1^r > \sigma_2^r$; **Case 2B**: $c_1 = c_2,\ \sigma_1^l > \sigma_2^l,\ \sigma_1^r < \sigma_2^r$.
- **Case 3A**: $c_1 < c_2,\ \sigma_1^l \geq \sigma_2^l,\ \sigma_1^r \leq \sigma_2^r$; **Case 3B**: $c_1 > c_2,\ \sigma_1^l \leq \sigma_2^l,\ \sigma_1^r \geq \sigma_2^r$.
- **Case 4A**: $c_1 < c_2,\ \sigma_1^l \geq \sigma_2^l,\ \sigma_1^r > \sigma_2^r$; **Case 4B**: $c_1 > c_2,\ \sigma_1^l \leq \sigma_2^l,\ \sigma_1^r < \sigma_2^r$.
- **Case 5A**: $c_1 < c_2,\ \sigma_1^l < \sigma_2^l,\ \sigma_1^r \leq \sigma_2^r$; **Case 5B**: $c_1 > c_2,\ \sigma_1^l > \sigma_2^l,\ \sigma_1^r \geq \sigma_2^r$.
- **Case 6A**: $c_1 < c_2,\ \sigma_1^l < \sigma_2^l,\ \sigma_1^r > \sigma_2^r$; **Case 6B**: $c_1 > c_2,\ \sigma_1^l > \sigma_2^l,\ \sigma_1^r < \sigma_2^r$.

VI. Forecasting and Results

The ASuPFuNIS was implemented with two sets of parameters. One with high learning rate and the other with a low learning rate. This gave us an idea of how fast a neural network based architecture is able to learn the pattern and make accurate forecasts.

Before training the network, the centers of the antecedents and the consequents fuzzy sets were initialized in the range (0,1.5). Both the feature and fuzzy weight spreads were intitalized in the range (0.2, 0.9). The learning rate and momentum values used in the experiment were the extremes, 0.2 and 0.02, 0.1 and 0.7 respectively. After the completion of the training phase, the test set was presented to the trained network. The performance criteria of ASuPFuNIS is the normalized root mean square error (NRMSE). NRMSE is defined as the ratio of the RMSE divided by the total number of patterns presented to the network.

The table III show the one step ahead forecast for the experiment.

Learning Rate	**0.2**	**0.02**
Momentum	0.1	0.1
Initial Min	Min normalized value of the data	Min normalized value of the data
Initial Max	Max normalized value of the data	Max normalized value of the data
No. of Rules	5	5

TABLE II
Experiment Set

(Learning Rate, Momentum)	(0.2,0.1)	(0.2,0.7)	(0.02,0.1)	(0.02,0.7)
One step ahead Forecast	38	44	49	47

TABLE III
Experimental Details

Acknowledgment

The authors would like to thank Prof. Satish Kumar, Head and Professor, Department of Physics and Computer Science for his constant guidance and support all through the work. We would also like to thank Dr. Lotika Singh, Assistant Professor in the same department at Dayalbagh Educational Institute, Dayalbagh, Agra, India.

References

[1] A. Sfetsos and C. Siriopoulos, "Time series forecasting of averaged data with efficient use of information," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 35, pp. 738–745, Sep. 2005.

[2] G. E. P. and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*. Holden-Day, 1970.

[3] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic Theory and Applications*. Prentice-Hall, 1995.

[4] Y.-K. Beng and C.-H. Lee, "Fuzzy time series prediction with data preprocessing and error compensation based on correlation analysis," in *Proc. IEEE Third 2008 International Conference on Convergence and Hybrid Information Technology*.

[5] Y. M. Y. and W. X. D., "Chaotic time series using least squares support vector machines," *Chinese Physics*, vol. 13.

[6] L. H. C., Z. J. S., and X. X. C., "Neural volterra filter for chaotic series prediction," *Chinese Physics*, vol. 14.

[7] M. J. and T. J. F., "Predict chaotic time-series using unscented kalman filter," *International Conference on Machine Learning and Cybernetics, Shanghai, China.*

[8] C. T. Lin and C. S. G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Upper Saddle River, NJ: Prentice-Hall, 1996.

[9] S. Pal and S. Mitra, *Neuro-fuzzy Pattern Recognition: Methods in Soft Computing*. New York: Wiley, 1999.

[10] J. Buckley and T. Feuring, *Fuzzy and Neural: Interactions and applications*, ser. Studies in Fuzziness and Soft Computing. Heidelberg, Germany: Physica-Verlag, 1999.

[11] S. K. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets and classification," *IEEE Trans. Neural Networks*, vol. 3, pp. 683–697, September 1992.

[12] H. Ishibuchi and Y. Hayashi, "A learning algorithm of fuzzy neural networks with triangular fuzzy weights," *Fuzzy Sets Syst.*, vol. 71, pp. 277–293, 1995.

[13] N. Kasabov, *Neuro-Fuzzy Techniques for Intelligent Information Processing*, ser. Studies in Fuzziness and Soft Computing. Heidelberg, Germany: Physica Verlag, 1999, vol. 30.

[14] J. S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst. Man, Cybern.*, vol. 23, pp. 665–685, May 1993.

[15] D. Nauck and R. Kruse, "A neuro-fuzzy method to learn fuzzy classification rules from data," *Fuzzy Sets Syst.*, vol. 89, pp. 277–288, 1997.

[16] C. F. Juang, "A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 10, pp. 155–170, April 2002.

[17] D. Chakraborty and N. R. Pal, "Integrated feature analysis and fuzzy rule-based system identification in a neuro-fuzzy paradigm," *IEEE Trans. Syst. Man, Cybern.*, vol. 31, pp. 391–400, 2001.

[18] S. Paul and S. Kumar, "Subsethood product fuzzy neural inference system (SuPFuNIS)," *IEEE Trans. Neural Networks*, vol. 13, no. 3, pp. 578–599, May 2002.

[19] H. Ishibuchi, "Neural network that learn from fuzzy if then rules," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 85–97, May 1993.

[20] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*. Upper Saddle River, NJ: Prentice Hall, 2001.

[21] S. Mitra and S. Pal, "Fuzzy multi-layer perceptron, inferencing and rule generation," *IEEE Trans. Neural Networks*, vol. 6, pp. 51–63, January 1995.

[22] M. Russo, "FuGeNeSys—A fuzzy genetic neural system for fuzzy modeling," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 373–388, August 1998.

[23] G. Box, G. M. Jenkins, and G. Reinsel, *Time Series Analysis - Forecasting and Control*. Prentice Hall, 1994.

[24] K. W. Hipel and A. I. McLeod, *Time Series Modelling of Water Resources and Environmental Systems*. Amsterdam, The Netherland: Elsevier, 1994.

[25] S. Kumar and C. S. Velayutham, *Asymmetric Subsethood-Product Fuzzy Neural Inference System (ASuPFuNIS)*. IEEE Transactions on Neural Networks, 2005.

[26] G. C. Mouzouris and J. M. Mendel, "Nonsingleton fuzzy logic systems: Theory and application," *IEEE Tran. Fuzzy Systems*, no. 1, pp. 56–71, February 1997.

[27] B. Kosko, *Fuzzy Engineering*. Englewood Cliffs: Prentice Hall, 1997.