Neural Comput & Applic (2002)11:90–102

Neural Computing & Applications

Forecasting Financial Time Series using Neural Network and Fuzzy System-based Techniques

V. Kodogiannis[1] and A. Lolis[2]

[1]Mechatronics Group, Department of Computer Science, University of Westminster, London, UK; [2]Ford Motor Company, R&D Center, Dunton, Essex, UK

Forecasting currency exchange rates are an important financial problem that is receiving increasing attention, especially because of its intrinsic difficulty and practical applications. During the last few years, a number of nonlinear models have been proposed for obtaining accurate prediction results, in an attempt to ameliorate the performance of the traditional linear approaches. Among them, neural network models have been used with encouraging results. This paper presents improved neural network and fuzzy models used for exchange rate prediction. Several approaches, including multi-layer perceptions, radial basis functions, dynamic neural networks and neuro-fuzzy systems, have been proposed and discussed. Their performances for one-step and multiple step ahead predictions have been evaluated through a study, using real exchange daily rate values of the US Dollar vs. British Pound.

Keywords: Exchange rates; Finance; Forecasting; Neural networks; Neuro-fuzzy systems

1. Introduction

An estimation problem of particular importance in the field of financial engineering is the problem of forecasting or predicting trends in the foreign exchange market. The forecasting of exchange rates is actually a very difficult task because of the many correlated factors that become involved. These factors could be economic, political and even psychological. Thousands of academic researchers and business practitioners have developed many types of forecasting methods in an attempt to find a reliable explanation of the movement of exchange rates. All these methods could be categorised into two broad classes:

- the fundamental analysis, and
- the technical analysis.

The former and most powerful approach depends upon exact knowledge of the various factors that influence the economy. When this knowledge is expressed in terms of precise Eqs, which can in principle be solved, it is possible to predict the future behaviour of the system. The main problem with this approach is that knowledge of the rules governing the behaviour of the system is not readily available. On the other hand, the second and less powerful method for prediction relies on the discovery of strong empirical regularities by analysing a set of past data. There are problems, however, with the latter approach, as regularities are not always evident, to be masked with noise.

Many techniques have been proposed in the last few decades for exchange rate prediction. The traditional statistical forecasting methods have relied on linear models such as the Box-Jenkins method, the exponential smoothing and the autoregression (AR) method. The Box-Jenkins method [1] requires the autocorrelation function for identifying proper autoregressive integrated moving average models (ARIMA). However, a major obstacle of this technique is its slow performance. The AR model is used to describe the stochastic behaviour of a system, and assumes that the rate at a specific time period can

Correspondence and offprint requests to: Dr V. Kodogiannis, Mechatronics Group, Department of Computer Science, University of Westminster, London HAI 3TP, UK. Email: kodogiv @wmin.ac.uk

be estimated by a linear combination of the previous time periods. Generally, the larger the data set, the better is the result in terms of accuracy, though with an increase in computational cost. Exponential smoothing is a convenient way of expressing the forecast in terms of exponentially smoothed statistics. The main problem with it is that it performs piece-wise linear approximation, and finds it quite difficult to model 'volatile' time series. An additional obstacle is that *a priori* estimates of the degree of nonlinearity of time series are required to select the order of smoothing. In practice, however, this is not always readily available. The drawbacks of the linear methods, as well as the development of artificial intelligence, have led to the development of alternative solutions utilising nonlinear modelling.

Two of the forecasting techniques that allow for the detection and modelling of nonlinear data are rule induction and neural networks, as described in Kingdom [2]. Rule induction identifies patterns in the data and expresses them as rules. Expert systems are an example using this technique. The effectiveness of this method, however, depends upon the quality of the attributes used in classification, and suffers from a number of drawbacks. First, the algorithm upon which rule induction is based produces a decision tree, which is difficult to interpret. Secondly, it is aimed at analysing small data sets. Thirdly, it fails to extract all the knowledge from the data; and finally, it presumes the existence of an expert capable of making accurate forecasts that will train the system. However, it is extremely difficult to transform the knowledge of an expert to mathematical rules.

Fuzzy logic systems are a generalisation of the rule-based systems. One of the most active research areas on the modelling and identification of nonlinear systems using fuzzy logic systems is in the fields of financial forecasting [3]. A Fuzzy Logic System (FLS) is a combination of linguistic variables and a set of IF-THEN rules using fuzzy logic principles [4]. It is a system that utilises fuzzy set theory and its operations. The most commonly used fuzzy logic system consists of a fuzzifier, inference engine, fuzzy rule base and defuzzifier. An Adaptive Fuzzy Logic System (AFLS) can be defined as the fuzzy logic system whose rules are extracted from numerical data through training, i.e. a FLS equipped with training algorithms so that all its parameters (e.g. centres, spreads) can be adapted in the same manner as with neural networks. AFLSs have been proved to be universal function approximators [5].

Li developed a fuzzy learning system which combined expert knowledge and machine learning to achieve competent performance in various applications. However, the results presented are questionable, as during the learning period the proposed system had not been capable of approximating accurately the real data set [6]. In an alternative approach, Pellizzari concluded that fuzzy logic techniques could give satisfactory results only when the volatility of the data is low. Real exchange rates between the US dollar and the Italian Lira have been used, but some crisis periods where the rate changed in an unpredictable descending way have been removed without giving sufficient explanation. This fact makes his proposed approach unsuitable for use in real exchange rate prediction, where volatility is extremely high and periods of crisis are the most important to be predicted [7]. Ghroshray [8] developed a system based on fuzzy regression, and tried to show its effectiveness in exchange rate prediction. It was proved that the prediction based on real data is not efficient for linear modelling, especially in the case when the market is moving in an unpredictable way.

Neural Networks (NNs) have recently gained popularity as an emerging and challenging computational technology, and they offer a new avenue to explore the dynamics of a variety of financial applications. NNs can simulate fundamental and technical analysis methods using fundamental and technical indicators as inputs. Consumer price index, foreign serve, GDP, export and import volume, etc. could be used as inputs. For technical methods, the delayed time series data, moving average, relative strength index, etc. could be used as inputs. The main focus of this research is forecasting as an example of function estimation and approximation. However, NNs as models for forecasting exchange rates, have been investigated in a number of previous studies [9,10]. Research conducted by Yao and Poh [11] has showed that without the use of extensive market data or knowledge, useful prediction can be done and significant paper profit can be achieved by utilising a simple NN with three layers. His proposed network was trained with the Basic backpropagation (BP) algorithm, and was used to predict one-step ahead, weekly exchange rates between the US dollar and five other major currencies. The output results were compared with those taken using the ARIMA model. Focusing on the gradients, the ARIMA methods can achieve 50% of correctness, while up to 73% can be achieved using neural network models. Refenes et al. [12] described a nontrivial application in forecasting exchange rates, and its implementation using a Multi Layer Perceptron (MLP) network. They showed that with careful network design, the BP learning procedure is an effective way of training three-layered neural

networks for time series prediction. The results taken from the various NNs for one-step ahead predictions were quite satisfactory, and outperformed all the linear models. On the other hand, for multiple step a-head prediction, the networks had a poor performance, mainly due to the inherent structure of the MLP.

The main characteristic of the studies mentioned above is the use of a simple three-layered MLP using the basic BP algorithm for training. Additionally, weekly data have been used, although it is well known that such data contains substantially less noise and is less volatile than real daily data. Researchers striving to develop a method to outperform NNs have tried to analyse time series data using the chaos theory. In many previous studies, chaotic models have been used for exchange rate prediction [13,14]. Lisi et al. [15] compared chaotic models vs. NNs, and found that the latter performed slightly better. However, as in the previous cases-studies, monthly data have been used.

In this paper, algorithms that follow the time series approach are been considered. In this case, NNs trace previous 'currency' patterns and predict a 'currency' pattern using recent data. The datasets of two currencies studied in this research comprise 1000 daily rates from the end of 1997 to the end of March 2000. For the purposes of this study, the first 800 values were reserved from a total of 1000 as training patterns, while the last 200 were the testing ones. Figure 1 illustrates these 'currency' patterns. For a univariate time-series forecasting problem, the inputs of the network are the past lagged observations of the data series, and the outputs are the future values. Each input pattern is composed of a moving window of fixed length along the series. The proposed network is a mapping function of the form:

$$y_{t+1} = F(y_t, y_{t-1}, \cdots, y_{t-p}) \tag{1}$$

where y_t is the observation at time t and p is the dimension of the input vector or the number of past observations related to the future value. In this sense, the feedforward network used for time series forecasting is a general AR model. The balance of this paper contains a comparative study of various prediction techniques used to develop a forecasting tool for exchange rate prediction.

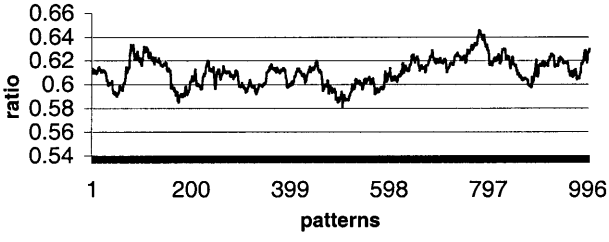

Fig. 1. Exchange rates $–£ for the period 1997–2000.

2. Nonlinear Modelling

2.1. Multi-layer Perceptron Architectures

Some artificial neural network architectures exhibit the capability of forming complex mappings between input and output that enable the network to approximate general nonlinear mathematical functions. The MLP neural network, trained by the standard BP algorithm, is probably the most widely used network, and its mathematical properties for nonlinear function approximation are well documented [16]. The generalised delta rule is applied for adjusting the weights of the feedforward networks to minimise a predetermined cost error function. The rule of adjusting weights is given by the Eq.

$$w_{ij}^p(t+1) = w_{ij}^p(t) + \eta\delta_j^p y_j^p + \alpha\Delta w_{ij}^p(t) \tag{2}$$

where η is the learning rate parameter, α the momentum term, and δ is the negative derivative of the total square error with respect to the neuron's output. To provide sufficient information for modelling using an MLP, a structure with two hidden layers and five inputs was used.

An alternative representation, called Spread Encoding (SE), has been shown to enable a network to maintain a higher degree of accuracy [16]. In the SE technique, each data value is represented as the mean value of a sliding Gaussian pattern of excitation over several nodes at the network input and output. A similar and reverse procedure is applied at the network output to decode the output back into the original variable range. This approach has similarities with data fuzzification techniques, where the scalar dimensional space of each variable is fuzzified to a space of higher dimensions. Also, decoding of the network output using the SE method consists of computing a weighted summation of the node excitations that is analogous to the conventional centre of gravity defuzzification technique. Thus, a network utilising spread encoding can be considered as a fuzzy-neural-type network. The SE method is also in keeping with the heritage of neural networks from biological systems, where information is often represented by the combined activity of a population of receptors, as in the retina of the eye. Figure 2 illustrates the internal architecture of this technique. Analytically, this data conditioning method of SE consists of mapping each network variable, $x \in [x_{\min}, x_{\max}]$, onto a sliding Gaussian

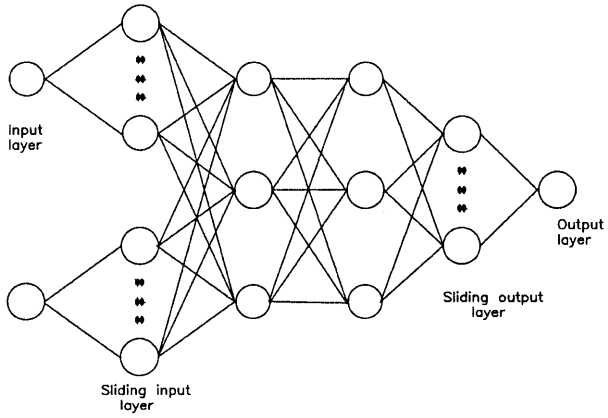

Fig. 2. Spread encoding neural architecture.

activation pattern of N network nodes, which includes additional nodes at either side of the variable range to contain overspill resulting from the use of a mapping function with wide support. The level of activation of each node is confined to be in the range [0.1, 0.9], similar to the conventional normalisation technique. Each node is assigned a value, α_i, linearly spaced by a distance, δ, to span the range of x, and the centre of the Gaussian excitation pattern corresponds to the value coded [17].

The SE algorithm is derived by creating a discrete map which represents the mean value of a continuous probability distribution, $\varphi(\alpha)$, within each class interval. This then provides a simple mechanism for retrieving the original coded value as a sum of the activity of the node excitations, each weighted by the values at the centres of the class intervals α_i. For a particular value of x the excitation of each node is defined by

$$\psi_i(x) = \frac{\int_{\alpha_i-\delta/2}^{\alpha_i+\delta/2} \alpha\varphi(\alpha - x)d\alpha}{\alpha_i} \tag{3}$$

which satisfies the requirement that

$$\sum_{i=1}^{N} \alpha_i\psi_i(x) = \int \alpha\phi(\alpha-x)d\alpha = \bar{\alpha} = x \tag{4}$$

It is assumed that the distribution $\varphi(\alpha)$ has unit area. The activation of a particular node can be evaluated from Eq. (3) by integration by parts:

$$\alpha_i\psi_i(x) = [\alpha\Phi(\alpha - x)]_{\alpha_i-\delta/2}^{\alpha_i+\delta/2} - \int_{\alpha_i-\delta/2}^{\alpha_i+\delta/2} \Phi(\alpha - x)d\alpha \tag{5}$$

where $\Phi(\alpha)$ is a parent cumulative distribution with $\varphi(\alpha) = \Phi'(\alpha)$. In the investigations reported in this paper, the integral term in Eq. (5) was approximated using the first two terms in the trapezium rule, resulting in

$$\psi_i(x) \approx \Phi(\alpha_i + \delta/2 - x) - \Phi(\alpha_i - \delta/2 - x) \tag{6}$$

which was found to provide sufficient accuracy in these studies. The relationship between this coding technique and conventional fuzzification techniques is illustrated by considering a first approximation of the integral term in Eq. (5) resulting from a Taylor series expansion of the cumulative function about the interval centre, α_i, and keeping the linear term in the expansion. This leads to

$$\psi_i(\mathrm{x}) \approx \phi(\alpha_i - \mathrm{x}) \tag{7}$$

which is analogous to the use of membership functions in fuzzy logic [18].

The practical advantage of spread encoding is that it leads to more accurate models using static feedforward neural networks than representing normalised physical variables using single nodes. The main reason for this is that signal noise is reduced in the spread encoding representations by suitable matching of the coding function with the interval width spanned by each node, exploiting hence the network's fault tolerance. The spread encoding algorithm was implemented by initially scaling the data to a normalised range where the original data range $r \in [r_{\min}, r_{\max}]$ was represented by $x \in [0, N-2N_0]$, with N the total number of nodes, and N_0 the number of nodes on either side of the variable range and $\delta=1$. The The data coding is performed using the following relation:

$$\psi_i(x) = \Phi(\alpha_i+1/2 - x) - \Phi(\alpha_i - 1/2 - x), \quad i = 1, \ldots N \tag{8}$$

where

$$\alpha_i = i - N_0 - c \tag{9}$$

and the cumulative Gaussian distribution function was approximated by the sigmoidal function centred at x:

$$\Phi(\alpha - x) = \frac{1}{1 + \mathrm{e}^{-\beta(\alpha-x)}} \tag{10}$$

In Eq. (9), c is an offset term that shifts the position of the range limits on the nodes. The width of the node excitations is inversely controlled by the parameter β in Eq. (10). Errors arise in decoding using a straightforward application of Eq. (4), because the node excitations, $\psi_i(x)$, are calculated by an approximation, Eq. (6). The accuracy of

decoding is improved by dividing the weighted sum by the sum of the node excitations. Thus, the network output is decoded back to the normalised range using

$$x = \frac{\sum_{i=1}^{N} \alpha_i \psi_i(x)}{\sum_{i=1}^{N} \psi_i(x)} \tag{11}$$

that is analogous to the conventional centre of gravity defuzzification technique.

In this study, the parameters used in the spread encoding algorithm were $N=6$, $N_0=2$, $c=0.7$ and $\beta=2.9$, which were found to provide sufficiently accurate coding and decoding in the application reported. Formal techniques for determining an optimum number of nodes in the hidden layers are still an area of current research, and presently, this task is often achieved by experimentation. The resulting MLP network topology with the spread encoding applied to the network data was 24 input nodes, 34 and 18 nodes for the two hidden layers and six output nodes.

2.2. Radial Basis Functions

An alternative model to multilayer networks for the time series identification is a neural network employing Radial Basis Functions (RBFs). An RBF is a function which has an in-built distance criterion with respect to a centre. A typical RBF neural network consists of three layers (input, hidden, output). The activation of a hidden neuron is determined in two steps: the first is computing the distance (usually by using the Euclidean norm) between the input vector and a centre c_i that represents the ith hidden neuron. Secondly, a function h that is usually bell-shaped is applied, using the distance obtained, to get the final activation of the hidden neuron. In our case, the well known Gaussian function $G(x)$

$$G(x) = \exp\left(-\frac{x^2}{\sigma^2}\right) \tag{12}$$

was used. The parameter σ is called 'unit width', and is determined using the heuristic rule *global first nearest-neighbour* [19]. It uses the uniform average width for all units using the Euclidean distance in the input space between each unit m and its nearest neighbour n. All the widths in the network are fixed to the same value σ, and this results in a simpler training strategy. The activation of a neuron in the output layer is determined by a linear combination of the fixed nonlinear basis functions, i.e.

$$F^*(x) = \sum_{i=1}^{M} w_i \phi_i(x) \tag{13}$$

where $\phi_i(x) = G(\|x - c_i\|)$ and w_i are the adjustable weights that link the output nodes with the appropriate hidden neurons. These weights in the output layer can then be learnt using the least-squares method.

The present study adopts a systematic approach to the problem of centre selection. Because a fixed centre corresponds to a given regressor in a linear regression model, the selection of RBF centres can be regarded as a problem of subset selection. The Orthogonal Least Squares (OLS) method can be employed as a forward selection procedure that constructs RBF networks in a rational way. The algorithm chooses appropriate RBF centres one by one from training data points until a satisfactory network is obtained. Each selected centre minimises the increment to the explained variance of the desired output, and so ill-conditioning problems occurring frequently in random selection of centres can automatically be avoided. In contrast to most learning algorithms, which can only work if a fixed network structure has first been specified, the OLS algorithm is a structural identification technique, where the centres and estimates of the corresponding weights can be simultaneously determined in a very efficient manner during learning. The OLS learning procedure generally produces an RBF network smaller than a randomly selected RBF network [20]. Due to its linear computational procedure at the output layer, the RBF is shorter in training time compared to its BP counterpart. A major drawback of this method is associated with the input space dimensionality. For large numbers of inputs units, the number of radial basis functions required can become excessive. If too many centres are used, the large number of parameters available in the regression procedure will cause the network to be over sensitive to the details of the particular training set and result in poor generalisation performance (overfit). To avoid this problem, an efficient combination of the zero-order regularisation and the OLS algorithm proposed by Chen et al. [21]. Similarly, the new error criterion for minimisation in the ROLS algorithm is

$$J = e^T e + \lambda g^T g \tag{14}$$

where g is the orthogonal weight vector which satisfies the triangular system

$$g = AW \tag{15}$$

and A is an upper triangular matrix computed

directly from the OLS regression procedure [20]. In this current study, the ROLS algorithm was employed to model the exchange rate problem. Best results were obtained using five inputs. The proposed 'global first nearest neighbour' method for width definition was found in practice to be inadequate for our problem. A rather heuristic method by taking the half the maximum Euclidean distance was finally adopted for our simulations. Although an elegant approach to the selection of the regularisation parameter λ is to adopt Bayesian techniques, in this work this parameter was set by trial and error to small positive values (i.e. 0.0003), which satisfy the optimal problem's solution.

2.3. Autoregressive Recurrent Neural Network

In the previous sections, the so-called *windowed input network* has been applied for modelling the exchange rate of US$ vs. GB£. Another approach has been that of explicitly including the time depedency into the network structure. The commonly used BP algorithm contains no *memory*, hindering the learning of temporal patterns. Here, the alternative is to use a dynamic network that is given some kind of memory to encode past history. In the standard MLP structure, the input to a neuron are multiplied by feedforward weights and summed, along with a node bias term. The sum is then passed through a smooth sigmoidal transfer function, producing the neuron's output value. This neural model has no memory, because the output value is not expilicitly dependent upon previous outputs. Recurrent neural networks have important capabilities such as attractor dynamics and the ability to store information for later use. However, the fully connected recurrent network, where all neurons are coupled to one another, is difficult to train and to make it converge in a short time.

A new model called the autoregressive recurrent network (ARNN), which can converge in reasonable training time, is proposed, and a generalised BP algorithm is developed to train the ARNN. The idea is that is still uses a recurrent neural network model, but the recurrent neurons are decoupled so that each neuron only feeds back to itself. With this modification, the ARNN model is considered to converge easier, and to need fewer training cycles than the fully recurrent network [22]. The ARNN is a hybrid feedforward/feedback neural network, with the feedback represented by recurrent connections appropriate for approximating the dynamic system. The structure of the ARNN is shown in Fig. 3. There are two hidden layers, with sigmoidal transfer functions, and a single linear output node. The ARNN topology allows recurrency only in the first hidden layer. For this layer, the memory-less BP model has been extended to include an autoregressive memory, a form of self-feedback where the output also depends upon a weighted sum of previous outputs. A modified BP algorithm is developed to train the ARNN [17]. The mathematical definition of the ARNN is shown below:

$$y(t) = O(t) = \sum_{1} W_l^O \, Q_l(t), \; Q_1 \qquad (16)$$
$$= f(S_l), \; S_1 = \sum_{j} W_{jl}^H Z_j(t)$$

and

$$Z_j(t) = f(H_j(t)), \; H_j(t) = \sum_{k=1}^{k=n} W_{jk}^D Z_j(t-k) \qquad (17)$$
$$+ \sum_{i} W_{ij}^I I_i$$

where $I_i(t)$ is the ith input to ARNN, $H_j(t)$ is the sum of inputs to the jth recurrent neuron in the first hidden layer, $Z_j(t)$ is the output of the jth recurrent neuron, $S_l(t)$ is the sum of inputs to the lth neuron in the second hidden layer, $Q_l(t)$ is the output of the lth neuron in the second hidden layer, and $O(t)$ is the output of the ARNN. Here, $f(\bullet)$ is the sigmoid function and W^I, W^D W^H and W^O are input, recurrent, hidden and output weights, respectively. The memories in each node at the first hidden layer allow the network to encode state information. The ARNN was trained as a prediction model for currency exchange using a structure of 4/16/8/1 nodes.

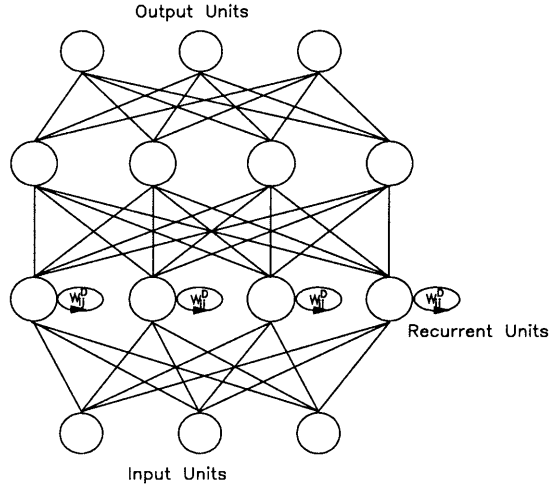

Fig. 3. ARNN architecture.

2.4. Elman Network

Recurrent networks with feedback exhibit a dynamic behaviour incorporating a temporal aspect not well conveyed by feedforward networks. They also create their own state variables and delays, thus requiring less information about the system being modelled. As an alternative to the ARNN architecture, the recurrent network developed by Elman has a simple architecture, and it can be trained using the standard BP learning algorithm. The context units of the Elman network memorise some past states of the hidden units, so the output of the network depends upon an aggregate of the previous states and the current input. This is the reason why the Elman network possesses the characteristic of a dynamic memory. In this architecture, in addition to the input, hidden and output units, there are also context units. The input and output units interact with the outside environment, while the hidden and context units do not. The input units are only buffer units that pass the signals without changing them. The output units are linear units which sum the signals fed to them. The hidden units have nonlinear sigmoidal functions. The context units are used only to memorise the previous activations of the hidden units, and therefore can be considered to function as one-step time delays. The feedforward connections are modifiable, but the recurrent are fixed. This network has been proved to be effective for modelling nonlinear systems not higher than the first order [23]. For this reason, an idea based on the work of Hertz et al. [24] was employed to configure a modified Elman network that is shown in Fig. 4. Here, self-connections are introduced in the context units of the network in order to give these units a certain amount of inertia. The introduction of self-feedback in the context units increases the possibility of the Elman network to model high-order systems [17]. Thus the output of the jth context unit in the modified Elman network (M.ELMAN) is given by

$$x_{cj}(t+1) = \alpha x_{cj}(t) + x_j(t) \tag{18}$$

It can be shown that

$$x_{cj}(t+1) = x_j(t) + \alpha x_j(t-1) + \alpha^2 x_j(t-2) + \ldots \tag{19}$$

Usually, α is between 0 and 1. A value of α nearer to 1 enables the network to trace further back into the past.

In this section, we use this network architecture for the load-forecasting problem. For this purpose, the output of the jth context unit in the modified Elman network structure is given by

$$x_{cj}(t+1) = x_j(t) + \alpha x_j(t-1) + \alpha^2 x_j(t-2) + \alpha^3 x_j(t-3) + \alpha^4 x_j(t-4) + \alpha^5 x_j(t-5) \tag{20}$$

The five 'memories' in each node at the context layer allow the network to encode state information [25]. To enhance the network's performance, an extra hidden layer has been added, and the linear output function was replaced with a standard sigmoidal one. Therefore, a 4/16/24/1 Modified Elman network was applied with self-feedback in the 16 context units, with α equal to 0.25.

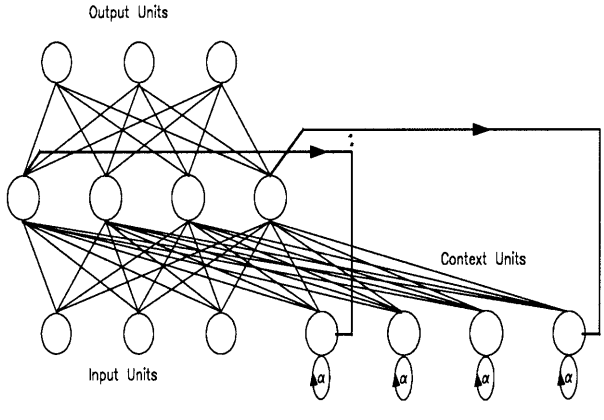

Fig. 4. Modified Elman architecture.

2.5. Neuro-Fuzzy Inference System

The various neural architectures presented in the previous sections illustrated their strength for modelling the forecasting problem. It is well known that a number of complex systems are difficult to model due to their nonlinear behaviour and imprecise measurement information. Therefore, imprecise systems states and a set of imprecise heuristic rules are needed. Zadeh introduced the linguistic approach to system design based on fuzzy logic. The main goal of a fuzzy inference system is to model human decision making within the conceptual framework of fuzzy logic and approximate reasoning [18]. Such a system consists of four important parts: the fuzzifications interface, knowledge base unit, decision-making unit, and output defuzzification interface.

Recently, the resurgence of interest in the field of NNs has injected a new driving force into the 'fuzzy' literature. The BP learning rule, which drew little attention until its application to NNS was discovered, is actually an universal learning paradigm for any smooth parameterised model, including fuzzy inference systems. As a result, a fuzzy inference system can now not only take linguistic information (linguistic rules) from human experts, but

also adapt it, using numerical data (input/output pairs) to achieve better performance. This gives fuzzy inference systems an edge over neural networks, which cannot take linguistic information directly. In this section a simple fuzzy logic system implemented by using a multilayer feedforward NN is presented for the prediction of foreign exchange rates. A schematic diagram of the proposed fuzzy neural network 'N-Fuzzy' structure is shown in Fig. 5.

The system consists of four layers. Nodes in layer one are *input nodes*, that represent input linguistic variables. Nodes in layer two are *membership nodes*, that act like membership functions. Each membership node is responsible for mapping an input linguistic variable into a possibility distribution for that variable. The *rule nodes* reside in layer three. Taken together, all the layer three nodes form a fuzzy rule base. Layer four, the last layer, contains the *output variable nodes*. The links between the membership nodes and the rule nodes are the premise links, and those between the rule nodes and the output nodes are the consequence links. For each rule node, there is at most one premise link from a membership node of a linguistic variable. Hence there are $\Pi_i|T(x_i)|$ rule nodes in the proposed 'N-Fuzzy' structure. Here $|T(x_i)|$ denotes the number of fuzzy partitions of input linguistic variable x_i. Moreover, all consequence links are fully connected to the output nodes and interpreted directly as the strength of the output action. In this way, the consequence of a rule is simply the product of the rule node output, which is the firing strength of the fuzzy rule and the consequence link. Thus, the overall network output is treated as a linear combination of the consequences of all rules instead of the complex composition, a rule of inference and the defuzzification process [26].

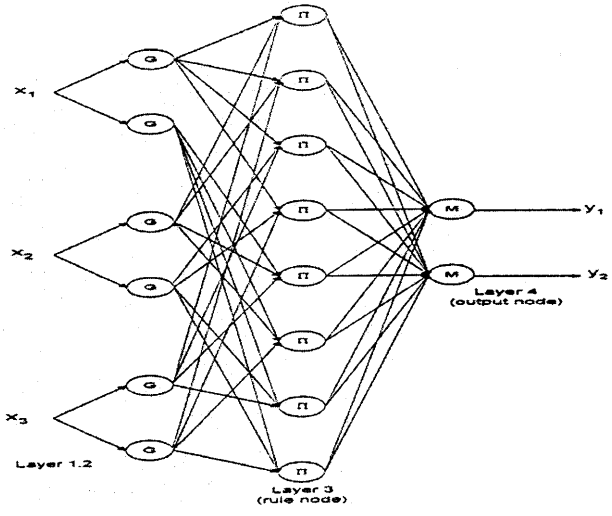

Fig. 5. 'N-Fuzzy' architecture.

For an n-input, one-output system, let x_i be the ith input linguistic variable and a^j the firing strength of rule j, which is obtained from the product of the grades of the membership functions $\mu_{A_i^j}(\mathrm{x}_i)$ in the premise part. If w^j represents the jth consequence link weight, then the inferred value y^* is obtained from the weighted sum of its inputs. Thus, the inference in the proposed fuzzy neural network is realising as

$$j\text{th rule: IF } x_1 \text{ is } A_1^j, \ldots, \mathrm{x}_n \text{ is } A_n^j, \text{ then } y = w^j, j = 1, 2, \ldots, m \tag{21}$$

$$y^* = \sum_{j=1}^{m} a^j w^j, \quad a^j = \prod_{i=1}^{n} \mu_A{}^{\xi_i}(x_i)$$

The class of fuzzy inference systems under consideration is a simplified type that uses a singleton to represent the output fuzzy set of each fuzzy logical rule. Thus, w^j is the consequence singleton of the jth rule. We now consider the basic function of each node in each layer.

Layer 1: layer 1 is an input layer whose nodes represent input variables. The nodes just transmit input values to the next layer directly. Hence, for the jth node of layer 1, the net input and output are represented respectively as

$$\mathrm{net}_j^1 = x_i^1, \quad i = j, y_j^1 = \mathrm{net}_j^1 \tag{22}$$

Layer 2: this is an input term (partition) layer whose nodes represent the membership functions associated with each linguistic term of the input variable. The Gaussian function, a particular example of radial basis functions, is adopted here as a membership function. Hence, the output of the jth term node associated with x_i is

$$\mathrm{net}_j^2 = \mu_{A_{ij}}(\mathrm{m}_{ij}, \sigma_{ij}) = -\frac{(\mathrm{x}_i^2 - \mathrm{m}_{ij})^2}{(\sigma_{ij})^2}, \quad \mathrm{y}_j^2 = \exp(net_j^2) \tag{23}$$

where m_{ij} and σ_{ij} are, respectively, the mean and the variance of the Gaussian function in the jth term of the ith input linguistic variable x_i^2.

Layer 3: layer 3 is a rule node layer, where each node represents a rule of the fuzzy system. Thus the nodes in Layer 3 form a rule base. The links in this layer are used to implement the antecedent matching. The matching operations or the fuzzy AND aggregation operation is chosen as the simple PRODUCT operation, instead of the MIN operation. Thus, for the jth rule node

$$\text{net}_j^3 = \prod_i^n \text{x}_i^3, \quad \text{y}_j^3 = \text{net}_j^3 \tag{24}$$

Layer 4: this is an output layer, whose nodes represent the partitions of the output variables. All consequence links are fully connected to the output nodes and interpreted directly as the strength of the output action. This layer performs centroid defuzzification to obtain the numerical output:

$$\text{net}_j^4 = \sum_i^n w_{ij}^4 x_i^4, \quad y_j^4 = \text{net}_j^4 \tag{25}$$

where the link weight w_{ij}^4 is the output action strength of the jth output associated with the ith rule. Thus, the overall net output is treated as a linear combination of the consequences of all rules, instead of the complex composition of a rule of inference and the defuzzification process.

The adjustment of the parameters in the proposed 'N-Fuzzy' system can be divided into two tasks, corresponding to the IF (premise) part and THEN (consequence) part of the fuzzy logical rules. In the premise part, we need to initialise the centre and width for Gaussian functions. To determine these initial terms, a Self-Organisation Map (SOM) and fuzzy-c-means algorithm are commonly used. Another simple and intuitive method of doing this is to use normal fuzzy sets to fully cover the input space. Since the final performance will depend mainly upon supervised learning, normal fuzzy sets are chosen for this work. In the consequence part, the parameters are output singletons. These singletons are initialised with small random values, as in a pure neural network. For our problem, four inputs were selected. Each input variable was assigned to four fuzzy partitions. The main advantages of the proposed method are the ability to learn from experience and a high computation rate. The average percentage relative error approaches its optimal value after a few epoch training. This is due to the fact that the consequence parameters have converged. This implies that the convergence of consequence parameters play a dominant role in system estimation accuracy. The remaining time is just for fine-tuning the premise parameters. Thus, the training required to achieve acceptable accuracy was very fast compared to the other techniques.

2.6. Adaptive Fuzzy Logic System (AFLS)

The development of a Fuzzy Logic System (FLS) can be done in a trial-and-error style where the initial properties of the FLS are specified by a general implementation procedure known as the straightforward fuzzy partitioning technique [27]. This implementation is usually done regardless of the distribution of data in input/output space and characteristic of the system at hand. However, this is computationally expensive and time consuming. A more promising approach to FLS is the adoption of an Adaptive Fuzzy Logic System (AFLS), i.e. a system having adaptive rules. Its structure is the same as a normal FLS, but its rules are derived and extracted from given training data. In other words, its parameters can be trained like a neural network approach, but with its structure in a fuzzy logic system structure. Since we have general ideas about the structure and effect of each rule, it is straightforward to effectively initialise each rule. This is a tremendous advantage of AFLS over its NN counterpart. The AFLS is one type of FLS with a singleton fuzzifier and a defuzzifier. The centroid defuzzifier cannot be used because of its computation expense, and as it prohibits using the error BP-training algorithm. The proposed AFLS consists of a new defuzzification approach, Balance Of Area (BOA). This AFLS has the same approach as the system presented by Wang [4], and its feed-forward structure is shown in Fig. 6 with an extra 'fuzzy basis' layer. For this particular forecasting problem, six input parameters proved to be adequate for modelling this financial time-series. The fuzzy basis layer consists of fuzzy basis nodes for each rule. A fuzzy basis node has the following form:

$$\phi_m(\bar{x}) = \frac{\mu_m(\bar{x})}{\sum_{l=1}^{L} \mu_l(\bar{x})} \tag{26}$$

where $\phi_m(\bar{x})$ is a fuzzy basis node for rule m and $\mu_m(\bar{x})$ is a membership value of rule m. Since we use a product-inference, the fuzzy basis node $\mu_m(\bar{x})$ is in the following form:

$$m(\bar{x}) = \prod_{i=1}^{n} \mu_{F_i^m}(x_i) \tag{27}$$

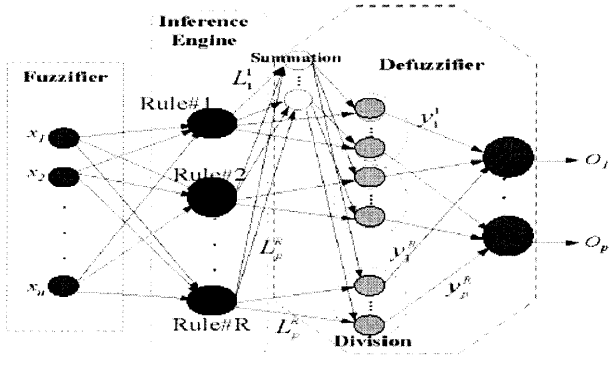

Fig. 6. AFLS architecture.

where $\mu_{F_i^m}(x_i)$ is a membership value of the ith input of rule m. In our case, a Gaussian shape as a membership function of each input of each rule has been used, hence $\mu_{F_i^m}(x_i)$ will be in the following form:

$$\mu_{F_{ii}^m}(x_i) = \exp\left[-\frac{(x_i - c_i^m)^2}{2(b_i^m)^2}\right] \tag{28}$$

where c_i^m *and* b_i^m are the centre and spread parameters, respectively, of the membership function ith input of the mth rule. The most popular defuzzification methods are the Centroid Of Area (COA) and Centre Average (CA). The former, although more accurate than the latter, is well known for its computational cost. Centroid calculation returns the centroid of the area formed by the consequent membership function, the membership value of its rules and the max-min or max product inference. However, since the COA method provides good performance, its main characteristics, centre of gravity and use of the shape of membership function will be preserved in the design of a new defuzzification approach. The overall output of the system may be the result of fuzzy union or the addition of rule outputs, as in Kosko's method. The proposed AFLS uses Kosko's method with product inference [28]. In general form, the calculation of the output, y, will be

$$y_p = \frac{\sum_{m=1}^{M} \mu_m L_p^m y_p^m}{\sum_{m=1}^{M} \mu_m L_p^m} \tag{29}$$

where y_p is the pth output of the network, μ_m is the membership value of the mth rule, L_p^m is the spread parameter of the membership function in the consequent part of the pth output of the mth rule, and y_p^m is the centre of the membership function in the consequent part of the pth output of the mth rule. For both Gaussian (and also triangular) shaped membership functions, the BOA defuzzifier gives results closer to the COA's than other defuzzification methods. This defuzzification approach can also be adaptive like AFLS with CA defuzzification and neural networks, and by using a BP technique, the update Eq.s can be derived [29]. Figure 7 illustrates the one-step ahead forecast for the exchange rate prediction of US$ and GBP£.

3. Results

Several structures of neural networks with algorithms ranging from simple MLP structures to neuro-

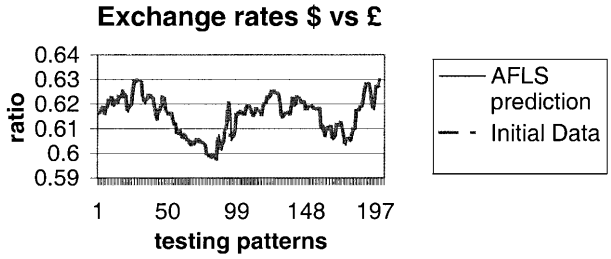

Fig. 7. One-step ahead prediction using AFLS-BOA system.

fuzzy ones were tested. The results and the statistics of forecasts obtained from application of the developed neural and fuzzy models for the one-and multi-step ahead exchange rate prediction of US$ and GBP£ [29] are given.

There is no consensus on the most appropriate measure to assess the performance of a forecasting technique. Three specific indices, namely SED, PRE and RMSE, have been used to evaluate the predictive performance of the proposed systems. These forecasting accuracy measures are listed as follows:

Standard Error Deviation (SED)

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}[y_i - \hat{y}_i]^2}$$

Percent Relative Error (PRE)

$$\epsilon = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \,.\, 100/y_i$$

Root Mean Square Error (RMSE)

$$\epsilon_i = \sqrt{\frac{1}{N}\sum_{i=1}^{N}[(y_i - \hat{y}_i\}/y_i]^2} \,.\, 100$$

These criteria are mean-based, and are frequently used performance measures in the literature. The complete one-step ahead results for the exchange rate prediction problem of forecast are illustrated in Table 1. For such an application, the widely used standard MLP with BP learning algorithm was considered in this work as a test-bed case. In an alternative representation, the SE structure has been shown to enable a network to maintain a higher degree of accuracy compared with the classic MLP structure. Although this considerable improvement in performance is generally at the expense of a larger network, the use of the proposed structure has significant advantages in applications requiring long-range predictions [25]. The performance of a classical MLP will severely deteriorate when it is acting as a recurrent model, because any errors of the predicted output will tend to accumulate. This problem is partly avoided in this specific structure by splitting

Table 1. One-step ahead forecasting results.

1-step	BP	SE	RBF	ARNN	ELM	N-Fuzzy	AFLS/gain over BP
PRE	0.31	0.2927	0.2646	0.2711	0.2683	0.272016	0.2567/+20.76%
RMSE	0.4215	0.4201	0.3905	0.4021	0.4010	0.39774	0.391
SDE	0.002597	0.0025	0.0023	0.0024	0.0024	0.002444	0.0024

the error into several nodes, thus exploiting the network's fault tolerance.

An alternative to the MLPs method could be a neural model employing radial basis functions. Both aspects of training time and improved accuracy were satisfied with the use of ROLS. An additional advantage of the specific algorithm was the overfitting problem avoidance by using the regularised parameter λ. Due to this factor, the RBF network enjoys a very compact structure compared to the other proposed neural architectures.

The use of dynamic neural networks presents an innovation to the specific problem. Here, the objective was to use two dynamic networks (ARNN and M.ELMAN) that were given some kind of memory to encode past history, with the additional requirements of short training time. Compared to the standard MLP structures, the improved results reveal the advantages of using memory neuron structures. The inclusion of memories and the related recurrence in the first hidden layer enable the network to carry out accurate predictions. Although this method is dependent on the number of 'memories' in the 'recurrent' nodes, and therefore it can be considered as a partially recurrent network, it proved to be the one with the fastest in training time. From the simulation results, it can be seen that the nonlinear system identification using ARNN and M.ELMAN are very promising. On the other hand, the reduced number of training cycles makes this technique a strong candidate for use in financial predictions.

However, the introduction of hybrid learning algorithms imposed a new dimension to this specific problem. The main advantages of the proposed Neuro-Fuzzy (N-Fuzzy) and (especially) AFLS methods are the ability to learn from experience and a high computation rate. The average percentage relative error approaches its optimal value after a short time in training. This is due to the fact that the consequence parameters have converged. This implies that the convergence of consequence parameters play a dominant role in system estimation accuracy. The remaining time is just for fine-tuning the premise parameters. Thus, the training required to achieve acceptable accuracy was very fast compared to the other techniques. In modelling, AFLS was used because of its trainability and generalisation. A modified AFLS was developed in contrast to the standard 'Wang's' AFLSs. The system consists of the same components except a different defuzzifier, balance of area defuzzifier. The Balance Of Area (BOA) defuzzifier uses the shape information of fuzzy membership functions in the consequence part of the IF-THEN rules to obtain the result. Its output is close to the Centroid Of Area (COA) defuzzification while requiring much less computation. With these choices of components, the AFLS can be trained by several training algorithms, such as the BP or genetic algorithms. In this research, the AFLS developed has been trained by the standard BP algorithm. A graphical presentation of the Percent Relative Error (PRE) index for the various methods involved in the one-step ahead forecasting is illustrated in Fig. 8. Compared with the BP approach, the AFLS technique has an evident 20% improvement/gain.

As a next task, a *k*-step ahead prediction is performed. Although all systems have been trained in a one-step ahead mode, it is desirable to investigate their performance in a multi-step prediction. In this particular case, the predicted outputs were fed back into the networks to predict more values. As the number of steps ahead increases, it is expected that the prediction error variance should also increase. Table 2 summarises the performed results for four-step, eight-step and 12-step ahead predictions, while Fig. 9 illustrates the differences between various methods for the multiple step case. AFLS still outperforms the other techniques, having a positive gain over classical MLP using the BP algorithm. As can be seen in Table 2, a positive

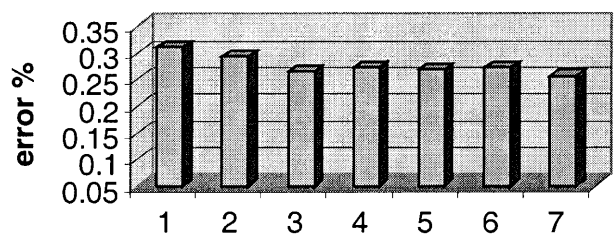

Fig. 8. One-step ahead comparison table.

Table 2. Multiple steps ahead forecasting results.

	k-steps	BP	ELM	N-Fuzzy	AFLS/gain over BP
PRE	4	0.7436	0.6558	0.6297	0.6258/+18.82%
	8	0.9266	0.8618	0.8042	0.7991/+15.95%
	12	1.038	1.0619	1.0025	0.9726/+6.724%
RMSE	4	0.8952	0.8505	0.8024	0.8067
	8	1.1348	1.0768	1.0498	1.0085
	12	1.3120	1.3196	1.2972	1.2597
SDE	4	0.005473	0.005249	0.004944	0.004979
	8	0.006927	0.006649	0.006463	0.006228
	12	0.008	0.008153	0.007987	0.007786

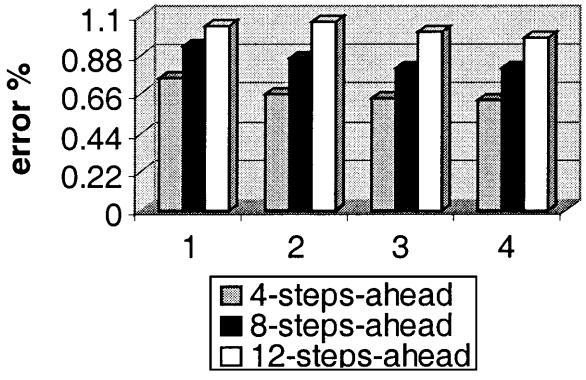

Fig. 9. *k*-steps ahead comparison table.

gain/improvement exists for the various steps. Figure 10 illustrates the results for eight-step ahead predictions using the AFLS technique. It seems, however, that above a 12-step prediction, a convergence of all techniques occurred. This may not be the result of poor network performance, however. Financial indexes, like exchange rates, can only be predicted accurately in the short-term using pure time-series. Using additional information, such as interest rates, oil prices, and stock exchange indexes, can enhance the present approach.

4. Conclusions

This study is based on the comparative analysis of neural network and fuzzy systems. These methods were developed for a one-step and multi-step ahead prediction of US$ and GBP£ daily exchange rates. Several neural architectures were tested, including MLPs, fuzzy-neural-type networks, radial basis and memory neuron networks. As an alternative to classic MLPs structures, this paper adopts the introduction of dynamics into the network by transferring the regular network neuron state to another set of *duplication* neurons called *memory neurons*. The autoregressive and M.ELMAN neural networks are examples of such an architecture. Their performance is characterised by a high degree of accuracy, as well as fast training time, similar to the performance of the RBF network based on the ROLS algorithm. The introduction of hybrid learning algorithms imposed a new dimension on the exchange rate prediction. The main advantages of the proposed N-fuzzy algorithm, and especially the AFLS, with the inclusion of an innovative defuzzification method are the ability to learn from experience and a high computation rate. In future work, the present approach will be enhanced by using advanced neuro-fuzzy models and additional information, such as interest rates, oil prices and stock exchange indexes.

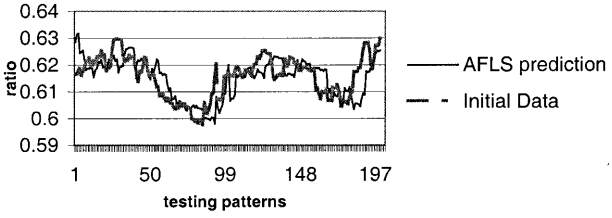

Fig. 10. Eight-steps-ahead prediction using AFLS-BOA system.

References

1. Box G, Jenkins G (1976) Time series analysis: forecasting and control. Holden-Day, San Francisco
2. Kingdom J (1997) Intelligent systems and financial forecasting. Springer-Verlag, Berlin
3. Bojadziev G, Bojadziev M (1997) Fuzzy logic for business, finance, and management. World Scientific
4. Wang LX (1994) Adaptive fuzzy systems and control. Prentice Hall
5. Wang LX (1992) Fuzzy systems are universal approximators. IEEE Int Conf on Fuzzy Systems San Diego, California, 1163–1170
6. Li T, Fang L, Guo D, Klasa S (1995) Predicting exchange rates using a fuzzy learning system. Computational Intelligence for Financial Engineering: Proc IEEE/IAFE, 103–107

7. Pellizzari P, Pizzi C (1997) Fuzzy weighted local approximation for financial time series modelling and forecasting. Computational Intelligence for Financial Engineering. Proc IEEE/IAFE, 137–143
8. Ghoshray S (1996) Application of fuzzy regression models to predict exchange rates for composite currencies. Computational Intelligence for Financial Engineering: Proc IEEE/IAFE, 264–270
9. Zhang G, Hu MY (1998) Neural network forecasting of the British Pound/US Dollar exchange rate. Omega Int J Manage Sci 26(4): 495–506
10. Zhang G, Patuwo B, Hu MY (2001) A simulation study of artificial neural networks for nonlinear time-series forecasting. Computers & Operations Research 28: 381–396
11. Yao J, Poh HL (1995) Forecasting the KLSE index using neural networks. Int Conf on Neural Networks 2: 1012–1017
12. Refenes AN, Azema-Barac M, Chen L, Karoussos SA (1993) Currency exchange rate prediction and neural network design strategies. Neural Comput & Applic, 46–58
13. Ghoshray S (1996) Foreign exchange rate prediction by fuzzy inferencing in deterministic chaos. Computational Intelligence for Financial Engineering: Proc IEEE/IAFE, 96–102
14. Garliauskas A (1999) Neural network chaos and computational algorithms of forecast in finance. IEEE Int Conf on Systems, Man and Cybernetics 2: 638–643
15. Lisi F, Schiavo RA (1999) A comparison between neural networks and chaotic models for exchange rate prediction. Computational Statistics & Data Analysis 30: 87–102
16. Rumelhart D, McClelland TL (eds) (1986) Parallel distributed processing: explorations in the microstructure of cognition. Vol. 1: Foundations. MIT Press
17. Kodogiannis VS (1994) Neural network techniques for modelling and learning control of an underwater robotic vehicle. PhD thesis, Liverpool University
18. Zadeh LA (1965) Fuzzy sets. Information and Control 8: 338–353
19. Moody J, Darken C (1989) Fast learning in networks of locally tuned processing units. Neural Computation 1: 281–294
20. Chen S, Cowan CFN, Grant PM (1991) Orthogonal least-squares algorithm for radial basis function networks. IEEE Trans Neural Networks 2(2): 302–309
21. Chen S, Chang ES, Alkadhimi K (1996) Regularised orthogonal least squares algorithm for constructing radial basis function networks. Int J Control 64(5): 829–837
22. Kodogiannis VS (2000) Comparison of advanced learning algorithms for short-term load forecasting. Intelligent & Fuzzy Systems 8(4): 243–261
23. Pham DT, Liu X (1993) Identification of linear and non-linear dynamics systems using recurrent neural networks. Artificial Intelligence in Engineering 8: 67–75
24. Hertz J, Krogh A, Palmer RG (1991) Introduction to theory of neural computation. Addison-Wesley
25. Kodogiannis VS, Lisboa PJG, Lucas J (1996) Neural network modelling and control for underwater vehicles. Artificial Intelligence in Engineering 1: 203–212
26. Kodogiannis VS, Anagnostakis EM (1999) A study of advanced learning algorithms for short-term load forecasting. Engineering Appl of Al 12(2): 159–173
27. Mendel J (2001) Uncertain rule-based fuzzy logic systems. Prentice Hall
28. Kosko B (1992) Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence. Prentice Hall
29. Lolis A (2000) A comparison between neural network and fuzzy system based techniques for exchange rate prediction. MSc dissertation, University of Greenwich, UK