Fuzzy Backpropagation Neural Networks for Nonstationary Data Prediction

Ramon Soto C.

Centro de Investigación y Desarrollo de Ingeniería Avanzada
Real #30 Col. Villa Satélite, Hermosillo, Sonora, Mexico
rsotocc@hotmail.com

Abstract. The backpropagation neural network is one of the most widely used connectionist model, especially in the solution of real life problems. The main reasons for the popularity of this model are its conceptual simplicity and its ability to tackle a broad range of problems. But, on the other hand, this architecture shows a well known problem for dealing with nonstationary data. In this paper, a variation of feedforward neural model which uses qualitative data both for feeding the network and for back propagating the error correction is presented. The data are coded by means of a fuzzy concept of local stability.

Keywords: Neurofuzzy modelling, data prediction.

1 Introduction

Soft computing paradigm is a useful way for tackling hard real problems. The main ingredients in most soft computing applications are artificial neural networks and fuzzy logic, being this the most successful combination of intelligent techniques [10].

An artificial neural network (ANN) is a computational model originally inspired in the bioelectrical networks formed in the brain. Several architectures have been proposed and implemented for realizing this metaphor, being the so called multilayer perceptron the best known and the one most extensively used. The multilayer perceptron or multiperceptron consists of multiple layers of processing units, typically interconnected in a feedforward way, what is called a feedforward neural network. Backpropagation algorithm, on the other hand, is the main technique for adjusting the synaptic weights of feedforward neural networks. In this case, the neural network uses to be called a backpropagation neural network (BNN).

The backpropagation neural network is one of the most widely used connectionist model, especially for the solution of real life problems. The main reasons for the popularity of this model are its conceptual simplicity and its ability to tackle a broad range of problems. But this architecture shows a well known problem for dealing with nonstationary data.

Fuzzy sets, on the other hand, are a useful way for representing imprecise concepts from real world. Opposing to classical sets, objects (or values) can belong to several fuzzy sets at the same time, even if those fuzzy sets represent contradictory concepts. Fuzzy sets have been shown to be a good tool for representing the qualitative categories describing the behavior of a dynamical system. Standard fuzzy sets have been shown to be a successful tool for modelling nonlinear systems [4], while second order fuzzy sets are a good approach for analyzing data with high noise levels [3]. But standard fuzzy sets give an atemporal description of world and do not represent the dynamical evolution of those categories. Temporal and dynamical fuzzy sets have been proposed for dealing with the dynamical nature of the world [5,9].

In this paper, a variation of feedforward neural model which uses fuzzy coded data, both for feeding the network and for back propagating the error correction, is presented. The coding is realized using soft dynamical fuzzy sets, in order of improve the network efficiency when dealing with nonstationary data.

2 Backpropagation Neural Networks

A *feedforward neuronal network* is a system integrated by a set of simple processing devices called nodes (neurons) organized in groups called *layers*, with the information flowing from the input layer to the output layer (figure 1). The neurons in each layer are interconnected only to the neurons of the following layer (from input to output). The output of each neuron is used as the input to the neurons of the next layer with which it is connected to. The output of the i-th neuron in any layer l is given by

$$x_i^{(l)} = g\left(a_i^{(l)}\right) \tag{1}$$

where, $a_i^{(l)}$ is called the neuron activity level and describes how much the input data influence the neuron activity. $a_i^{(l)}$ is defined as

$$a_i^{(l)} = \left(\sum_{j=1}^{n^{(l-1)}} w_{ij}^{(l)} x_j^{(l-1)}\right) - \Theta_i^{(l)} \tag{2}$$

$w_{ij}^{(l)}$ is the synaptic weight between the j-th neuron in layer $l-1$ and the i-th neuron in layer l; $n^{(l-1)}$ is the number of neurons in layer $l-1$; $x_j^{(l-1)}$ is the output of the j-th neuron in layer $l-1$ and $\Theta_i^{(l)}$ is the bias value of the i-th neuron in layer l. The output function $g(\cdot)$ used in this work is given by

$$g(a) = \frac{1}{1+e^{-Ga}} \tag{3}$$

being G the network gain.

The neural network training consists of adjusting the synaptic weights between the neurons, in such a way that the output of neurons in the output layer corresponds to a certain known value associated to the input data. Several

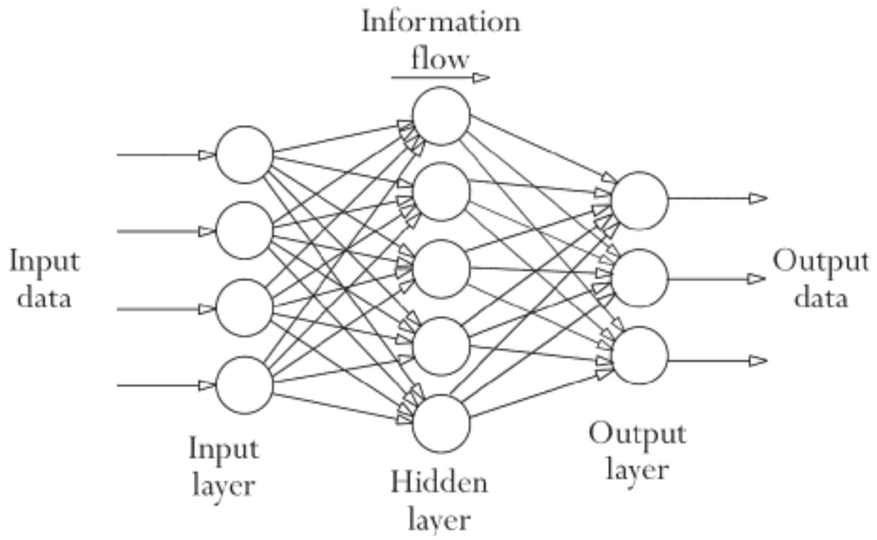

Fig. 1. Structure of a feedforward neural network

methods for training a feedforward neural network have been proposed, being the backpropagation algorithm the most extensively used.

In backpropagation algorithm, the output error is computed individually for each neuron in the output layer, and then this error is back propagated towards the first hidden layer.

The synaptic weight $w_{ij}^{(l)}$ that connects the jth neuron in layer $l-1$ to the i-th neuron in layer l, is iteratively updated by means of the equation

$$w_{ij}^{(l)}(k+1) = w_{ij}^{(l)}(k) + \varDelta w_{ij}^{(l)}(k) \tag{4}$$

where

$$\varDelta w_{ij}^{(l)}(k) = \lambda \delta_i^{(l)}(k) x_j^{(l-1)} + \alpha \varDelta w_{ij}^{(l)}(k-1) \tag{5}$$

$\lambda \in (0,1)$ is the learning ratio and $\alpha \in (0,1)$ the momentum. $\delta_i^{(l)}$ is the signal error of the i-th neuron in layer l and is defined as

$$\delta_i^{(l)} = \begin{cases} g'\left(a_i^{(l)}\right)\left(y_i - x_i^{(l)}\right) & \text{Neurons at output layer} \\ g'\left(a_i^{(l)}\right)\sum_{m=1}^{n^{(l+1)}} \delta_m^{(l+1)} w_{mi}^{(l+1)} & \text{Neurons at interior layers} \end{cases} \tag{6}$$

The derivative for a sigmoid output function is

$$g'(a) = Gg(a)\left(1 - g(a)\right) \tag{7}$$

A well known problem of backpropagation neural networks is its poor performance when dealing with nonstationary data. This limitation is evident when the behavior of the sigmoid function output is analyzed. From equation 3, can be observed that large activation values a lead to $g(a)$ values very close to the unit, whereas small values of activation lead to $g(a) \approx 0.5$ (see figure 2). This way, large input data values require small synaptic weight values at the input layer, while small input values need large weights, making the synaptic weights tuning a hard task. On the other hand, if the neural network output must be a value in the same range as the input data, then the output function must be attuned to throw values outside the range $[0, 1]$. A common way to solve these problems consists on normalizing the data, but this approach requires to know the complete range of values.

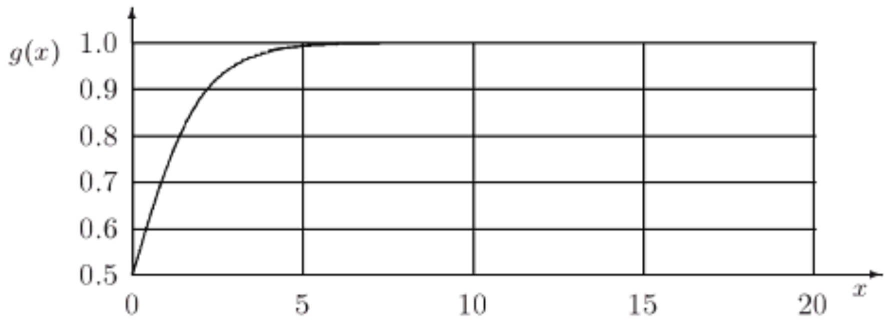

Fig. 2. Output function behavior for large activation values

3 Fuzzy Local Stability

Given a crisp universe of discourse U, a fuzzy set A over U is defined as

$$A = \{(x, \mu_A(x)) \mid x \in U,\ \mu_A : U \to [0, 1]\} \tag{8}$$

Where $\mu_A(x)$ is the *fixed* degree of membership of x in the fuzzy set A.

But, fuzzy categories corresponding to dynamical systems states can show a temporal evolution: The normal values for a given product price, the electric energy demand and the solar activity level, vary along time, in some cases showing some periodicity. If A corresponds to a dynamical category, it is natural to expect that the membership value of x in the fuzzy set A will change along any time interval T.

A dynamical fuzzy set [9] A_D is defined as

$$A_D = \left\{(x, \mu^{(x)}(t)) \mid \mu^{(x)}(t) \in [0, 1], t \in T\right\} \tag{9}$$

The behavior of the main variables in many complex dynamical systems can be described in terms of its local stability: the changes on the prices, for example, are say to be raising or dropping. Following this metaphor, the data generated by dynamical systems can be coded in terms of three dynamical fuzzy sets designated as Down, Stable and Up, as is shown in figure 3. Such fuzzy sets describe the local stability of the variable, related to its previous values, and are defined as follows:

– The Stable fuzzy set consist of a pair of increasing-decreasing logistical curves. The membership function for Stable fuzzy set is given by

$$\mu_S(x) = \begin{cases} 0 & x \leq \nu - \varpi \\ 2\left[\frac{\nu-\varpi-x}{\varpi}\right]^2 & \nu - \varpi < x \leq \nu - \varpi/2 \\ 1 - 2\left[\frac{\nu-x}{\varpi}\right]^2 & \nu - \varpi/2 < x \leq \nu + \varpi/2 \\ 2\left[\frac{\nu+\varpi-x}{\varpi}\right]^2 & \nu + \varpi/2 < x < \nu + \varpi \\ 0 & x \geq \nu + \varpi \end{cases} \qquad (10)$$

where $\mu_S(x)$ is the membership value of variable x in the Stable fuzzy set; ν is is the modal value of Stable fuzzy set and represents the reference value for the three fuzzy sets; ϖ is the fuzzy set bandwidth. In this work, the modal value is taken as the last value in the time series previous to the actual value, while the bandwidth value is chosen proportional to the standard deviation value over the adjacent differences on last 10 points in the time series previous to the actual value.

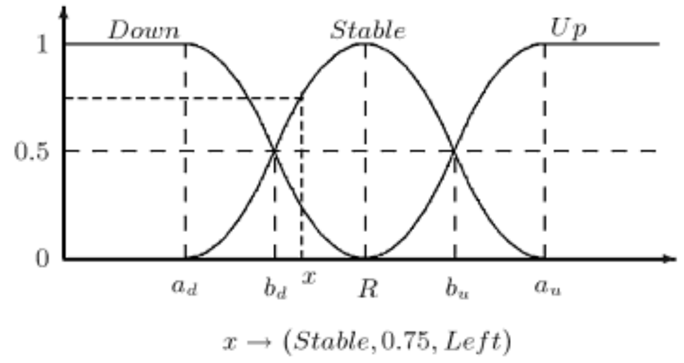

Fig. 3. Fuzzy sets describing local data stability

– The Down set is chosen as a decreasing logistical curve. The corresponding membership function is given by

$$\mu_D(x) = \begin{cases} 1 & x < \nu - \varpi \\ 1 - 2\left[\frac{\nu-\varpi-x}{\varpi}\right]^2 & \nu - \varpi \leq x < \nu - \varpi/2 \\ 2\left[\frac{\nu-x}{\varpi}\right]^2 & \nu - \varpi/2 \leq x < \nu \\ 0 & x \geq \nu \end{cases} \qquad (11)$$

where $\mu_D(x)$, is the membership value of variable x in the Down fuzzy set.

– The Up set is chosen as a growing logistical curve. This set is defined by the fuzzy membership function

$$\mu_U(x) = \begin{cases} 0 & x < \nu \\ 2\left[\frac{\nu-x}{\varpi}\right]^2 & \nu \leq x \leq \nu + \varpi/2 \\ 1 - 2\left[\frac{\nu+\varpi-x}{\varpi}\right]^2 & \nu + \varpi/2 \leq x \leq \nu + \varpi \\ 1 & x \geq \nu + \varpi \end{cases} \qquad (12)$$

where $\mu_U(x)$ is the membership value of variable x in the Up fuzzy set.

The fuzzy coding in this paper is based on the qualitative coding method used in the Fuzzy Inductive Reasoning (FIR) technique [1,2]. Such coding uses a triplet formed by a qualitative class value, a numeric membership value and an additional qualitative function side value (see figure 3). The last quantity specifies the membership function side where the corresponding quantitative value is located on. This is a suitable description for local dynamical stability: A variable declared as Stable with a Down tendency, can be coded as $(Stable, \mu, Left)$, for example.

In this coding scheme, a quantitative value is treated as belonging only to the fuzzy set where its membership value is larger. Such coding is equivalent but more economical and readable than the usual approach of coding a quantitative value based on its membership to several fuzzy sets. Two complementary concepts to this representation are those of "neighboring fuzzy sets" and "pointing out to". These concepts are defined as follows:

Definition 1. Two fuzzy sets α and β are said to be "neighboring fuzzy sets" iff $\alpha \neq \beta$ and $\alpha \cap \beta \neq \phi$, being ϕ the empty set. This relations is represented as $\mathcal{N}(\alpha, \beta)$.

Definition 2. The record r is said to "point out to" the fuzzy set α iff: 1) $r \in \beta$ and $\mathcal{N}(\alpha, \beta)$, and 2) α is located at the same side of β where r is placed on. This relations es represented as $r \mapsto \alpha$.

4 Fuzzy Backpropagation Neural Networks

A fuzzy backpropagation neural networks (FBNN) uses the same structure as that of standard feedforward neural network.

The input data for a fuzzy neural network are qualitative triplets $r = (Class$, μ, $Side)$. In order to maintain the data coherence, the neural network must operate over the three triplet elements. The output of the i-th neuron in layer l is given by

$$X_i^{(l)} = \left(\alpha_i^{(l)}, g(a_i^{(l)}), Side_i^{(l)}\right) \tag{13}$$

- The $\alpha_i^{(l)}$ value is obtained by aggregating the signals coming from the neurons of the previous layer connected to it, $l-1$. This is achieved by computing the next indices

$$I_{\alpha_s} = \sum_{j=1}^{n} w_{i,j}^{(l)} \cdot \chi_j \tag{14}$$

where $\alpha_s \in \{baja, estable, alza\}$, $w_{ij}^{(l)}$ is the synaptic weight between the j-th neuron in layer $l-1$ and the i-th neuron in the layer l. χ_j is computed by means of the following relation:

$$\chi_j = \begin{cases} g(a_j^{(l-1)}) & X_j^{(l-1)} \in \alpha_s \\ 1 - g(a_j^{(l-1)}) & X_j^{(l-1)} \in \beta, \mathcal{N}(beta, \alpha_s) \text{ y } X_j^{(l-1)} \mapsto \alpha_s \\ 0 & \text{a.o.c.} \end{cases} \tag{15}$$

$\alpha_i^{(l)}$ is chosen as the fuzzy set whose index gets the highest aggregation value.

– $g(\cdot)$ is the output function defined by equation [3], being

$$a_i^{(l)} = \max_s(I_{\alpha_s}) - \vartheta_i^{(l)} \tag{16}$$

$\vartheta_i^{(l)}$ is the bias value. Since the output membership value always lies in the range $[0.5, 1.0]$, $\vartheta_i^{(l)}$ is set to 0.5 for all neurons.

– The new function side value $Side_i^{(l)}$ is taken as pointing to the fuzzy set whose counter gets the second largest value in the aggregation process.

The error propagation is made using the same process as in the standard backpropagation algorithm, except because now the error must be computed from quelitative data. The error signal for i-th neuron in layer l, $\delta_i^{(l)}$, is defined as

$$\delta_i^{(l)} = \begin{cases} g'\left(a_i^{(l)}\right) \mathcal{D}(X_i^{(l)}, X_i) & \text{For neurons in the output layer.} \\ \\ g'\left(a_i^{(l)}\right) \sum_{m=1}^{n^{(l+1)}} \delta_m^{(l+1)} w_{mi}^{(l+1)} & \text{For neurons underneath the output layer.} \end{cases} \tag{17}$$

being $\mathcal{D}(\cdot)$ the soft Gower distance function, defined as

Definition 3. The soft Gower distance between two records, namely r_i and r_j is

$$\mathcal{D}(r_i, r_j) = \sqrt{\frac{1 - Gow(r_i, r_j)}{Gow(r_i, r_j)}} \tag{18}$$

X_i is the target value for the i-th neuron in the output layer and $X_i^{(l)}$ the actual output value at the same node. $Gow(r_i, r_j)$ is given by

$$Gow(r_i, r_j) = \frac{1}{3} \cdot (1 - |\mu_i - \mu_j| + g_c(r_i, r_j) + g_s(r_i, r_j)) \tag{19}$$

where

– μ is the membership value of the record r as given by the fuzzy triplet $(Class, \mu, Side)$.

– $g_c(r_i, r_j)$ is defined as follows:

$$g_c(r_i, r_j) = \begin{cases} 1 & r_i \in \alpha \text{ and } r_j \in \alpha \\ 0.5 & r_i \in \alpha, r_j \in \beta \text{ and } \mathcal{N}(\alpha, \beta) \\ 0 & \text{Any other case.} \end{cases} \tag{20}$$

being α and β fuzzy sets, and ϕ the empty set.

– And $g_s(r_i, r_j)$ is:

$$g_s(r_i, r_j) = \begin{cases} 1 & r_i \in \alpha, r_j \in \alpha \text{ and } S(r_i) = S(r_j) \\ 0.5 & r_i \in \alpha, r_j \in \beta, \mathcal{N}(\alpha, \beta), r_i \mapsto \beta \text{ and } r_j \mapsto \alpha \\ 0 & \text{Any other case.} \end{cases} \tag{21}$$

5 Results

The performance of FBNN is compared to that of standard BNN by tackling the following prediction problems:

– The Lorenz attractor, defined by the next differential equations system [8]:

$$\begin{aligned} \frac{dx(t)}{dt} &= \sigma(y - x) \\ \frac{dy(t)}{dt} &= rx - y - xz \\ \frac{dz(t)}{dt} &= xy - bz \end{aligned} \tag{22}$$

where x, y and z are state variables and r, σ and b are positive physical parameters. Following most simulations, the parameter values used in this work are $\sigma = 10$, $r = 28$ and $b = \frac{8}{3}$. The sampling rate is $\tau = 0.01$. A set of 3000 records were generated by numerical integration of equations 22 using a fourth order Runge-Kutta method. The test consist in predicting the z component of Lorenz system. This time series was labeled as "Lorenz 1" in table 1.

– The time series labeled as "Lorenz 2" in table 1 was generated by adding a linear growing tendency and a linear increasing dispersion to the component z of the Lorenz attractor. This series of given by the equation:

$$\breve{z}_k = \frac{k(1 + z_k)}{100}; \quad k = 1 \ldots 3000 \tag{23}$$

– A third time series was generated simulating a quadratic growing tendency and a quadratic increasing dispersion in the z component of the Lorenz attractor. This time series is labeled as "Lorenz 3" in table 1, and is given by the equation:

$$\tilde{z}_k = \frac{k^2(1 + z_k)}{50000}; \quad k = 1 \ldots 3000 \tag{24}$$

– The sunspots number. The data used in this work correspond to the observation range from January 1st 1947 to August 12 1956.

– Foreign currencies market. This problem is a well-known testbed for time series prediction methods. As other financial systems, this one presents many modeling problems [6,7]. The variables to be predicted in this study case are the exchange rates, related to the U.S. dollar, of the next currencies: the Pound Sterling, the Canadian Dollar, the German Mark, the Japanese Yen and the Swiss Franc. The data used in this problem were taken originally from the Monetary Yearbook of the Chicago Mercantile Exchange, and published in Internet by Andreas S. Weigend (http://www.cs.colorado.edu/∼andreas/Time-Series/Data/). This database contains 3512 records covering the time period from June 1st, 1973 to May 21st, 1987.

As can be observed in the table, the prediction of stationary time series by the fuzzy neuronal network is slightly better than the prediction obtained with the standard neuronal network. On the other hand, when the stationary conditions disappear, the performance of the fuzzy neuronal network is clearly superior.

In both cases a neuronal network with 15 input nodes, 20 nodes in the only hidden layer, and one unit in the output layer, was used. The training parameters for both neural networks were: gain=1, learning ratio=0.05, momentum=0.5. BNN models used bias=1 while FBNN used bias=0.5.

Table 1. NMSE error when predicting different time series with a standard backpropagation neural network and a fuzzy backpropagation neural network

	Neural network model	
Time series	**standard**	**fuzzy**
Lorenz 1	0.1202752	0.0960687
Lorenz 2: linear	0.3987142	0.0933912
Lorenz 3: quadratic	0.5365449	0.0869285
Sunspots number	0.2661172	0.2782048
Pound Sterling	0.3169981	0.0972515
Canadian Dollar	0.4199859	0.1927423
German Mark	0.2506006	0.0495888
Japanese Yen	0.3608898	0.0386783
Swiss Franc	0.1830959	0.0525080
Average NMSE	**0.3170246**	**0.1094846**

6 Conclusions

A fuzzy variant on backpropagation neural network has been presented. This model uses dynamical fuzzy sets for describing the local stability in nonstationary data. The performance of fuzzy backpropagation neural networks is compared the performance of standard backpropagation neural networks. The fuzzy variant seems to be clearly superior to the standard one.

References

1. F. Cellier. General system problem solving paradigm for qualitative modeling. In *Paul A. Fishwick and Paul A. Luker, editors, Qualitative Simulation Modeling and Analysis*, pages 51-71. Springer-Verlag, New York, 1991.
2. Á. de Albornoz. Inductive Reasoning and Reconstruction Analysis: Two Complementary Tools for Qualitative Fault Monitoring of Large-Scale Systems. PhD thesis, Universitat Politcnica de Catalunya, 1996.
3. N. N. Karnik and J. M. Mendel. Applications of type-2 fuzzy logic systems to forecasting of time-series. *Information Sciences*, 120:89–111, 1999.
4. I. Kim and S.-R. Lee. A fuzzy time series prediction method based on consecutive values. *In 1999 IEEE International Fuzzy Systems Conference Proceedings*, pages 703707, Seoul, Korea, 1999.
5. B.R. Kosanović, L.F. Chaparro, and R.J. Sclabassi. Signal analysis in fuzzy information space. *Fuzzy Sets and Systems*, 77(1):49–62, 1996.
6. C. Lee-Giles, S. Lawrence, and A. Chung-Tsoi. Noisy time series prediction using a recurrent neural network and grammatical inference. *Machine Learning*, 44(1/2):161–183, 2001.
7. M. Magdon-Ismail, A. Nicholson, and Y. Abu-Mostafa. Estimating model limitations in financial markets. *Proceedings of the IEEE*, 86(11), 1998.
8. S. Mukherjee, E. Osuna, and F. Girosi. Nonlinear prediction of chaotic time series using support vector machines. In J. Principe, L. Giles, N. Morgan, and E. Wilson, editors, (IEEE) *Workshop on Neural Networks for Signal Processing* (VII), page 511. IEEE Press, 1997.
9. R. Soto and G. Nunez. Dynamical fuzzy sets for time series forecasting. In *IASTED International Conference on Modelling and Simulation* (MS 2003), 2003.
10. A. Tsakonas and G. Dounias. Decision making on noisy time-series data under a neuro-genetic fuzzy rule-based system approach. In *Proceedings of 7th UK Workshop on Fuzzy Systems*, pages 80–89, 2000.