COMPARISON OF APPROXIMATION AND FORECASTIG ABILITY OF VARIOUS RBF NNW MODELS WITH STATISSTICAL MODELS

Milan Marček[1,2]

[1]Institute of Computer Science, Faculty of Philosophy and Science, The Silesian University Opava

[2]The Faculty of Management Science and Informatics, University of Žilina

Abstract

At first, we discuss the basic structure of the fuzzy system as a simple yet powerful fuzzy modeling technique. Neural networks and fuzzy logic models are based on very similar underlying mathematics. The similarity between RBF networks and fuzzy models is noted in detail. Then, we propose the extension of RBF neural networks by the cloud model. Time series approximation and prediction by applying RBF neural networks or fuzzy models and comparisons between the various types of RBF networks and statistical models are discussed

Key words: *Probabilistic time-series models, fuzzy system, classic and soft RBF network, cloud models, granular computing.*

1 Introduction

In this paper, we consider the approximation ability of ARMA models and models based on fuzzy systems to “explain” the behaviour of time-series variables. In addition, we explore some of the more important specifications associated with approximation of time-series variables using RBF networks.

2 A model of fuzzy systems

This section concentrates on the basic principles of identifying input-output functions of systems using fuzzy systems. Fuzzy systems theory have been recently consolidated and presented by B. Kosko [5].

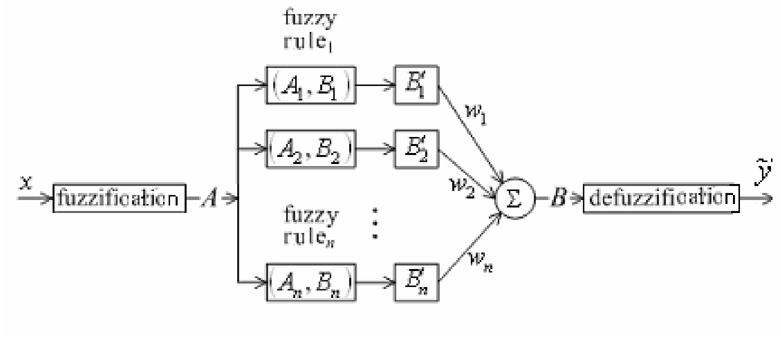

Fig. 1: Fuzzy system architecture.

The basic fuzzy system architecture is shown in Fig. 1. In this architecture the fuzzy system maps input fuzzy sets A to output fuzzy sets B. The fuzzy inference computes the output fuzzy sets B'_i, weights them with the weights w_i, and sums to produce the output fuzzy set B, i.e.

$$B = \sum_i w_i B'_i \qquad \textbf{(1)}$$

The fuzzy system is distributed and consists of a series of a separate fuzzy rules (relations) of the type of *if A_i then B_i*. Centroidal output converts fuzzy sets vector B to a scalar. The most popular centroidal defuzzification technique uses all the information in the fuzzy distribution B to compute the crisp y value as the centroid $\tilde{y}$ or centre of mass of B, i. e.

$$\tilde{y} = \int_{-\infty}^{\infty} y\mu_B(y)dy \, / \int_{-\infty}^{\infty} \mu_B(y)dy \tag{2}$$

where μ_B represents the union of all clipped output fuzzy sets. When the output membership functions are singletons, then, in the case of an $\Re^k \rightarrow \Re$ function, Eq. (2) becomes

$$\tilde{y} = \sum_{j=1}^{n} y_j \mu_j(x) / \sum_{j=1}^{n} \mu_j(x) \tag{3}$$

where y_j stands for the centre of gravidity of the *j*th output singleton, the notation μ is used for a membership function and *n* denotes the number of rules.

As mentioned earlier the output fuzzy sets can be calculated if all the separated fuzzy rules are known and the weights are determined. As in fuzzy logic systems all operations involve sets, the amount of calculation per inference rises dramatically. In a fuzzy system, powerful tools for generating fuzzy rules purely from data are neural networks. In next section we show, how to obtain fuzzy rules and how to determine the weights w_i for fuzzy system using RBF networks.

3 RBF neural network implementation of fuzzy logic

Fuzzy systems offer methodologies for managing uncertainty in a rule-based structure. In this section, RBF neural network structures are used (see Fig. 2) as tools of performing fuzzy logic inference for fuzzy system depicted in Fig. 1. We propose the neural architecture according to the Fig. 2 whereby the a priori knowledge of each rule is embedded directly into the weights of the network.

The structure of a neural network is defined by its processing units and their interconnections, activation functions, methods of learning and so on. In Fig. 2, each circle or node represents the neuron. This neural network consists an input layer with input vector $\mathbf{x}$ and an output layer with the output value $\hat{y}_t$. The layer between the input and output layers is normally referred to as the hidden layer. Here, the input layer is not treated as a layer of neural processing units. One important feature of RBF networks is the way how output signals are calculated in computational neurons. The output signals of the hidden layer are

$$o_j = \psi_2(\|\mathbf{x} - \mathbf{w}_j\|) \tag{4}$$

where $\mathbf{x}$ is a *k*-dimensional neural input vector, $\mathbf{w}_j$ represents the hidden layer weights, ψ_2 are radial basis (Gaussian) activation functions. Note that for an RBF network, the hidden layer weights $\mathbf{w}_j$ represent the centres $\mathbf{c}_j$ of activation functions ψ_2.

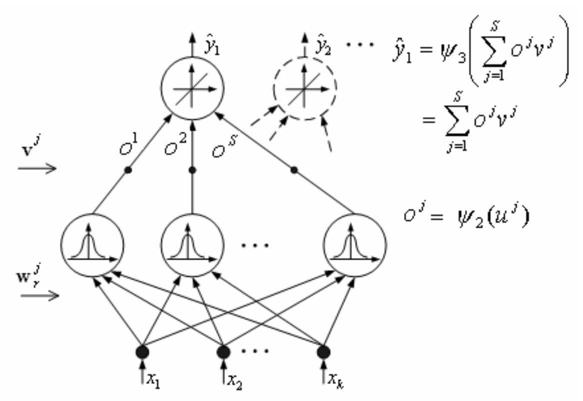

Fig. 2: RBF neural network architecture.

The output layer neuron is linear and has a scalar output given by $\hat{y} = \sum_{j=1}^{s} v_j o_j$ where v_j are the trainable weights connecting the component of the output vector $\mathbf{o}$. Then, the output of the hidden layer neurons are the radial basic functions of a proximity of weights and input values. A serious problem is how to determine the number of hidden layer (RBF) neurons. The most used selection method is to preprocess training (input) data by some clustering algorithm. After choosing the cluster centres, the shape parameters σ_j must be determined. These parameters express an overlapping measure of basis functions. For Gaussians, the standard deviations σ_j can be selected, i. e. $\sigma_j \sim \Delta c$ where Δc denotes the average distance among the centres.

To show the similarity of the RBF neural network and the fuzzy system, consider again the scalar output $\hat{y}$. The RBF network computes the output data set as

$$\hat{y}_t = G(\mathbf{x}_t, \mathbf{c}, \mathbf{v}) = \sum_{j=1}^{s} v_{j,t}\, \psi_2(\mathbf{x}_t, \mathbf{c}_j) = \sum_{j=1}^{s} v_j o_{j,t}, \qquad t = 1, 2, ..., N \tag{5}$$

where N is the size of data samples, s denotes the number of the hidden layer neurons. The hidden layer neurons receive the Euclidian distances ($\|\mathbf{x} - \mathbf{c}_j\|$) and compute the scalar values $o_{j,t}$ of the Gaussian function $\psi_2(\mathbf{x}_t, \mathbf{c}_j)$ that form the hidden layer output vector $\mathbf{o}_t$. Finally, the single linear output layer neuron computes the weighted sum of the Gaussian functions that form the output value of $\hat{y}_t$.

If the scalar output values $o_{j,t}$ from the hidden layer will be normalised, where the normalisation means that the sum of the outputs from the hidden layer is equal to 1, then the RBF network will compute the "normalised" output data set y_t as follows

$$y_t = G(\mathbf{x}_t, \mathbf{c}, \mathbf{v}) = \sum_{j=1}^{s} v_{j,t} \frac{o_{j,t}}{\sum_{j=1}^{s} o_{j,t}} = \sum_{j=1}^{s} v_{j,t} \frac{\psi_2(x_t, c_j)}{\sum_{j=1}^{s} \psi_2(x_t, c_j)}, \; t = 1, 2, ..., N. \tag{6}$$

The similarity of approximation schemes (6) and (3) is obvious. From these schemes is shown that the weights $v_{j,t}$ in Eq. (6) to be learned correspond to w_i in Eq. (1), and $\psi_2(./.)$ to $\mu_j(x)$ in Eq. (3). Thus, the adaptive fuzzy system depicted in Fig. 1 uses neural techniques to abstract fuzzy principles and to choose the weights w_i, and gradually refine those principles as the system samples new cases. These properties were firstly recognised by V. Kecman [3]. In Fig. 2, the network with one hidden layer and normalised output values $o_{j,t}$ is the fuzzy logic model or the soft RBF network.

Next, to improve the abstraction ability of soft RBF neural networks with architecture depicted in Fig. 2, we replaced the standard Gaussian activation (membership) function of RBF neurons with functions based on the normal cloud concept. Cloud models are described by three numerical characteristics [2]: Expectation (*Ex*) as most typical sample which represents a qualitative concept, Entropy (*En*) as the uncertainty measurement of the qualitative concept and Hyper Entropy (*He*) which represents the uncertain degree of entropy. *En* and *He* represent the granularity of the concept, because both the *En* and *He* not only represent fuzziness of the concept, but also randomness and their relations. This is very important, because in economics there are processes where the inherent uncertainty and randomness are associated with different time. Then, in the case of soft RBF network, the Gaussian membership function $\psi_2(./.)$ in Eq. (6) has the form

$$\psi_2(\mathbf{x}_t, \mathbf{c}_j) = \exp\left[-(\mathbf{x}_t - E(\mathbf{x}_j)/2(En')^2\right] = \exp\left[-(\mathbf{x}_t - \mathbf{c}_j)/2(En')^2\right] \tag{7}$$

where En' is a normally distributed random number with mean En and standard deviation He, E is the expectation operator.

4 An Application

We illustrate the classic, fuzzy logic (soft) and cloud (granular) RBF neural networks on the input – output function estimation of a sales process. The time plot of the data set used in this application (the 724 daily sales for Hansa Flex company, 2004-2005) is shown in Fig. 3.

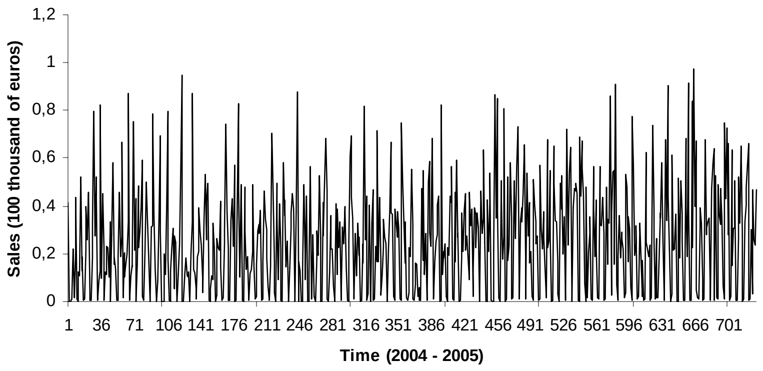

Fig. 3: Daily sales from January 2004 to December 2005

Statistical models chosen after some experimentation using the Statgraphics procedures were

$$y_t = \phi_1 y_{t-7} + \varepsilon_t \quad \text{or} \tag{8}$$

$$y_t - y_{t-7} = \theta_1 \varepsilon_{t-7} + \varepsilon_t \,. \tag{9}$$

Both statistical models have typical seasonal behavior with the seventh lag. Fitted models have the following forms: $\hat{y}_t = -0.1248 y_{t-7}$ or $y_t - y_{t-7} = 0.93868\varepsilon_{t-7}$ respectively. The usual diagnostic checking procedures according to Box & Jenkins [1] do not reveal any inadequacies in these models. The Box-Jenkins theory was also used to specify the neural input variables. As shown from Eq. (8) and (9), these variables are here y_{t-7} and ε_{t-7} respectively.

In the RBF neural network framework, the non-linear function $f(\mathbf{x})$ was estimated according to the expressions in Eq. (5). In the case of RBF fuzzy logic network, the non-linear input – output approximation function was estimated according to the formula (6). Next, the fuzzy logic RBF neural network was extended towards estimation with (a priori known) noise levels of the entropy. Noise levels are indicated by hyper entropy. It is assumed that the noise level is constant over time. We select, for practical reasons, that the noise level is a multiple, say 0.015, of entropy. In Table 1, we give the achieved results of approximation ability in dependence on various number of RBF neurons. The mean square error (MSE) was used to measure the approximation ability.

Table 1. The MSE´s measures of approximation accuracy of various RBF networks related to the different number of clusters (RBF neurons).

Numb. of RBF Neurons	NNW Archite-cture: Gausian Classic RBF	Soft RBF	Classic with Normal Cloud Concept	Soft with Normal Cloud Concept
RBF network representations for model (8):				
3	1.439	0.698	1.503	0.729
5	0.729	0.693	0.817	0.716
10	0.687	0.675	0.671	0.678
15	0.697	0.681	0.681	0.678
RBF network representations for model (9):				
3	0.783	0.646	0.786	0.647
5	0.810	0.632	0.803	0.630
10	0.607	0.571	0.607	0.571
15	0.582	0.563	0.582	0.563

The mean (centre), standard deviation of the clusters (RBF neurons) are computed using K-means algorithm. The data used are the same as used in the previous statistical models. As shown in Table 1, models that generate the "best" *MSE*′s are soft RBF networks.

Comparing both approaches, i. e. the models based on the Box-Jenkins methodology (the MSE for model expressed by Eq. (8) is 0.7793 and by Eq. (9) is 0.74606 respectively), and the models based on RBF networks approaches, we clearly see that models based on RBF networks are better approximation models because the estimated values are close to the actual values.

Table 2. The MSE´s measures of ex post forecast accuracy of various RBF networks related to the different number of clusters (RBF neurons).

Numb. of RBF Neurons	NNW Architecture:	Gausian Classic RBF	Soft RBF	Classic with Normal Cloud Concept	Soft with Normal Cloud Concept
RBF network representations for model (8):					
3		1.6634	0.8602	1.6092	0.8488
5		0.8509	0.8377	0.8489	0.8338
10		0.8055	0.8359	0.8051	0.8346
15		0.8433	0.8480	0.8391	0.8026
RBF network representations for model (9):					
3		0.8452	0.6869	0.8451	0.6879
5		0.8806	0.6548	0.8801	0.6549
10		0.6600	0.6241	0.6649	0.6245
15		0.6307	0.6248	0.8795	0.6052

Next, a forecast model was produced. Forecasts are provided during the ex post forecast period (y_{525}, …, y_{724}, i. e. the sample period ends with observation y_{524}). Table 2 presents the MSE´s measures of ex post forecast accuracy. As can be seen from Table 2, the soft RBF networks have indeed a forecasting power: if anything, it seems that they manage to forecast better than other RBF network architectures.

5 Conclusion

In this article, we have extended RBF neural network methodology to approximate the non-linear time series data using normal cloud models in the role of standard Gaussian activation (membership) function for RBF neurons. This was done by formulating a hyper entropy of standard deviation (entropy) of the Gaussian cloud model.

To approximate the input-output function of a business process, the RBF neural network approach was applied on the daily sales data of the Hansa Flex company and compared with an approach based on statistical procedures. For the sake of approximation abilities we evaluated 34 models. Two models are based on the Box-Jenkins time series analysis approach, and 32 models are based on the neural (fuzzy logic) methodology. Using the disposable data a very appropriate model is the soft RBF network with activation functions based on the granular concept. It is also interesting to note that the most computationally intensive models, the model based on the Box-Jenkins methodology, is newer considered "best".

Acknowledgement. This work was supported by the grant GAČR 402/08/0022.

References

[1] BOX, G. E. P. and JENKINS, G. M.: Time Series Analysis, Forecasting and Control. San Francisco, CA : Holden-Day, (1970

[2] CHANGYU LIU, DEYI LI, YI DU , XU HAN: Normal Cloud Models and Their Interpretation. The 11th World Congress of International Fuzzy Systems Association (IFSA 2005). Beijing China, July 28-31, 2005, Springer, Volume III, 1540–1543

[3] KECMAN, V: Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy logic Models. Massachusetts Institute of Technology, The MIT Press, (2001)

[4] KELLER, J., M., YAGER, R., R., TAHANI, H.: Neural network implementation of fuzzy logic. Fuzzy Sets and Systems 45 (1992), 1–12

[5] KOSKO, B.: Neural networks and fuzzy systems a dynamic approach to machine intelligence. Prentice Hall, Inc., 1992

[6] MARČEK, D.: Determination of fuzzy relations for economic fuzzy time series models by SCL techniques. The 11th World Congress of International Fuzzy Systems Association (IFSA 2005). Beijing China, July 28-31, 2005, Tsinghua University Press, Springer, Volume III, 1419–1424

[7] YOSHINARI, Z., PEDRITZ, W., HIROTA, K.: Construction of fuzzy models through clustering techniques. Fuzzy Sets and Systems 54 (1993), 157–165

Author:
Milan Marček:
e-mail: marcek@fria.utc.sk