Prediction of high frequency data applied to the Bond Found Time Series: Neural approach

Dusan Marcek [1]

Abstract

We develop forecasting models based on the neural approach for the forecasting of the bond price time series provided by the VUB bank and make their comparisons of the forecast accuracy with the class of the statistical ARCH-GARCH models. There is a limited statistical or computer science theory on how to design the architecture of the RBF networks for some specific nonlinear financial or economic time series, which allows the exhaustive study of underlying dynamics, and to determine their parameters. To illustrate the forecasting performance of these approaches the learning aspects of RBF networks are presented and the application is included. We show a new approach of function estimation for nonlinear time series forecasting model by means of granular neural network based on the Gaussian activation function and by incorporating a cloud concept in the RBF neural network. A comparative study shows that the presented approach is able to model and predict the high frequency data with reasonable accuracy and more efficiency than the statistical methods based on the ARCH-GARCH methodology.

Key words

Time series, statistical models, NN RBF type models, clustering, granulation

JEL Classification: C32, C45, C53, G32

1. Introduction

The aim of the paper is to provide information about the examination of the RBF NN (Radial BasicFunction Neural Network models for the forecasting of the bond price time series provided by the VUB bank and comparisons of the forecast accuracy with the class of statistical ARCH-GARCH type models. Our motivation for this comparative study lies in both the difficulty for constructing of appropriate statistical ARMA (AutoRegressive Moving Average) and ARCH-GARCH (AutoRegressive - Generalised Conditionally Heteroscedastic models (so called hard computing) to forecast volatility events in ex post simulations and the recently emerging problem-solving methods that exploit tolerance for imprecision to achieve low solution costs (soft computing).

Over the past few years, new information technologies based on the fact that financial time series contain nonlinearities have been developed. That documents competitive performance of neural networks on a larger number of time series. In this connection, we developed a new approach of function specification for time series. This function was specified by means of the granular RBF (Radial Basic Function) neural network (GNN) based on the Gaussian RBF and by incorporating a cloud concept in the RBF. For the sake of the evaluation, the Root Mean Squared Error (RMSE) measure is used to quantify the goodness of fit.

[1] Dusan Marcek Silesian University Opava, Faculty of Philosophy and Science, Czech Republic, e-mail: dusan.marcek@fpf.slu.

2. Granular comnputing

Granular computing is a separate research of field of artificial intelligence [1, 2, 3]. In 1979, Zadeh first introduced and discussed the notion of information granulation and suggested that fuzzy set theory may find potential applications in this respect, which pioneers the explicit study of granular computing [4]. A general framework of granular computing was given in a paper [5] by Zadeh based on fuzzy set theory. According to Zadeh, granules are constructed and defined based on the concept of generalized constraints. Relationships between granules are represented in terms of fuzzy graphs or fuzzy if-then rules. The associated computation method is known as computing with words.

The basic idea of granular computing, i.e. problem solving with different granularities, have been explored in many fields such as artificial intelligence, interval analysis, quantization, Dempster-Shafer theory of belief functions, rough set theory, cluster analysis, machine learning, databases and many others [6]. In the past few decades we witnessed rapid development of granular computing. Three branches of science, namely fuzzy theory, artificial intelligence and rough set theory contributed significantly to the study of granular computing. Basic property of a granule is its size. Intuitively, the size may be interpreted as the degree of detail, abstraction or concreteness. In the cluster analysis, the size of granule can be regarded as the number of elements in the cluster. Granules of a similar type may be collected together and their collective properties may be studied. An important property of granules and granular levels is the granularity. The granularity of a level refers to the collective properties with respect to their size. Granularities are reflected by the size of all granules involved in levels. It enables to construct a hierarchical structure called a hierarchy. The term hierarchy is used to denote a family of partially ordered granulated levels in which each level consists of a family of interacting and interrelated granules [6]. The basic function of granular computing is granulation. The granulation includes the process of construction the three basic components of granular computing: granules, granulated levels and hierarchies. There are two types of the granulation: top-down decomposition of large granules to smaller granules and the bottom-up combination of smaller granules into larger granules.

In this paper we will use clustering to find homogeneous groups of data and then represent them by granules. The granulation process is schematically depicted in Fig. 1. Granules extracted from the data are then described by the three digital characteristics of normal cloud model, namely expected value, entropy and hyper-entropy. The mean of a granule is regarded as the expected value of normal cloud model and the standard deviation of a granule provides the entropy of normal cloud model. Both characteristics are calculated in the process of competitive learning. The hyper-entropy of a granule is a measure of variance of the cloud drops and it can be calculated using backward algorithm [1, 3] or set up it manually. This granulation process is regarded as bottom-up granulation and will be used and more formally described in the next section.

3. Neural approach: Forecasting model based on G RBF NN

In this section we show a new approach of function estimation for time series modelled by means a granular RBF neural network based on Gaussian activation function modelled by cloud concept. We propose the neural architecture according to Fig. 2.

Fig. 1. Derivation of granules from input data and their description by normal cloud model

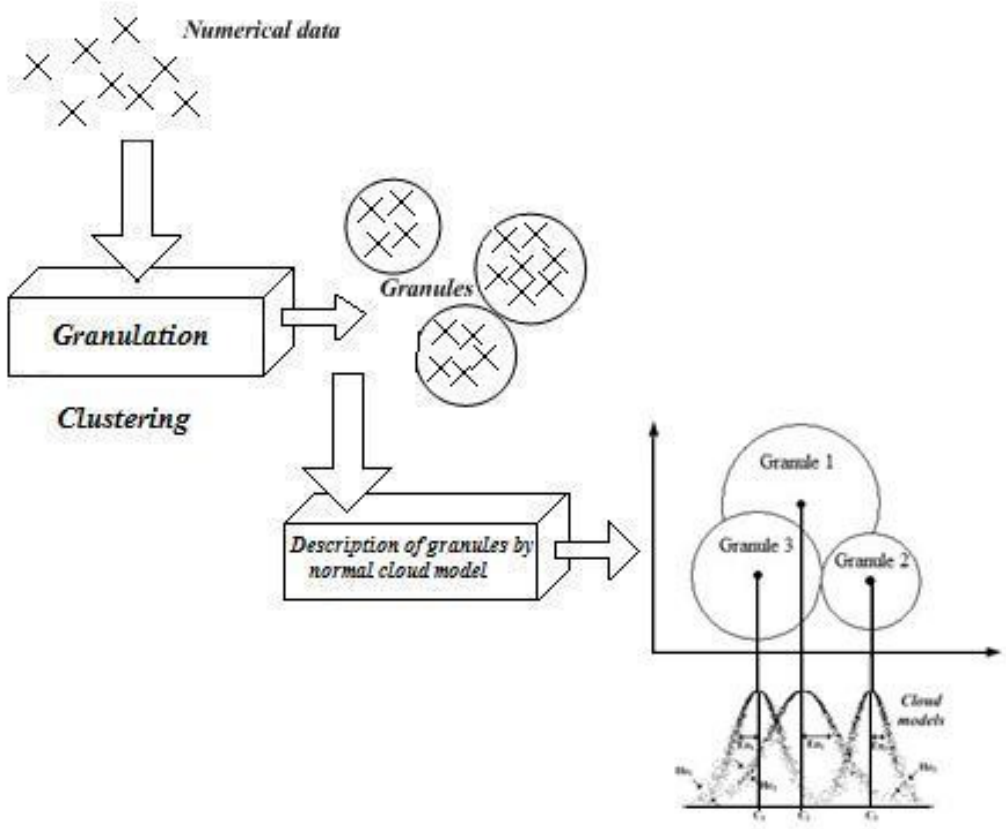
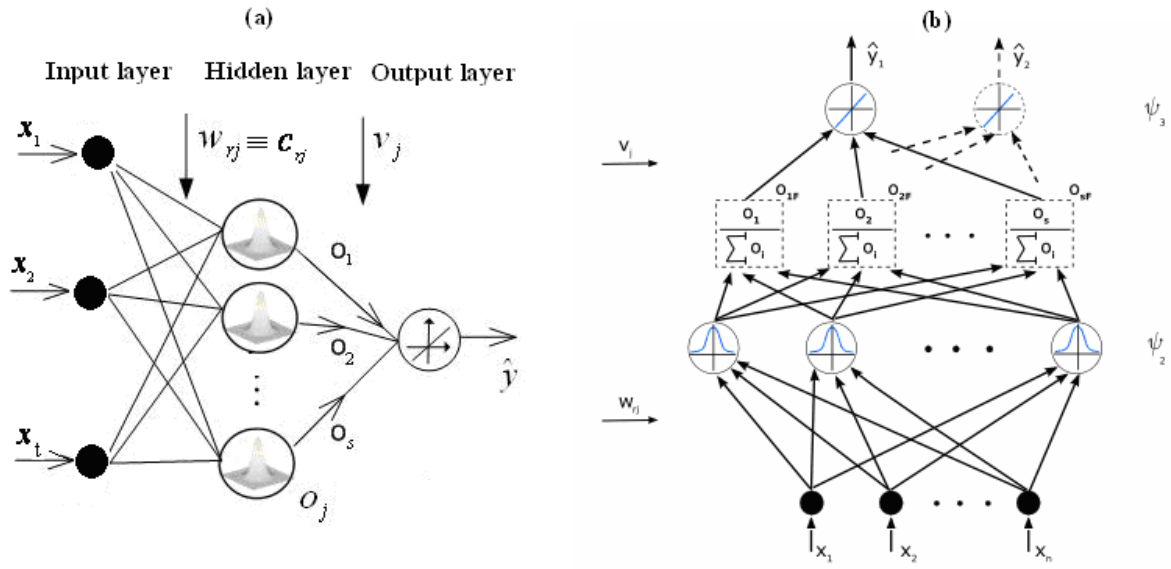


Fig. 2. RBF neural network architecture (a)classic, (b) fuzzy logic



The structure of a classic RBF NN is defined by its architecture (processing units and their interconnections, activation functions, methods of learning and so on). In Figure 2 each circle or node represents the neuron. This neural network consists an input layer with input vector **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 and its neurons as RBF neurons. 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

$$o_j = \psi_2(\|\mathbf{x} - \mathbf{w}_j\|) \tag{1}$$

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. To find the weights $\mathbf{w}_j$ or centres of activation functions we used the following adaptive (learning) version of *K*-means clustering algorithm for *s* clusters:

Step 1. Randomly initialise the centres of RBF neurons $\mathbf{c}_j^{(t)}$, $j = 1, 2, \ldots, s$ where *s* represents the number of chosen RBF neurons (clusters).

Step 2. Apply the new training vector $\mathbf{x}^{(t)} = (x_1, x_2, ..., x_k)$.

Step 3. Find the nearest centre to $\mathbf{x}^{(t)}$ and replace its position as follows

$$\mathbf{c}_j^{(t+1)} = \mathbf{c}_j^{(t)} + \lambda(t)\,(\mathbf{x}^{(t)} - \mathbf{c}_j^{(t)})$$

where $\lambda(t)$ is the learning coefficient and is selected as linearly decreasing function of *t* by $\lambda(t) = \lambda_0(t)\,(1 - t/N)$ where $\lambda_0(t)$ is the initial value, *t* is the present learning cycle and *N* is number of learning cycles.

Step 4. After chosen epochs number, terminate learning. Otherwise go to step 2.

The second parameter of the RB function, the standard deviation is estimated as K multiple, ($K \geq 1$), of the mean value of quadratic distance among the patterns and their cluster centres.

The output layer neuron is linear and has a scalar output given by

$$\hat{y} = \sum_{j=1}^{s} v_j o_j \tag{2}$$

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 the 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. $\sigma_j \sim \Delta c_s$, where Δc_s denotes the average distance among the centres.

The RBF network computes the output data set as

$$\hat{y}_t = G(\mathbf{x}_t, \mathbf{c}, \mathbf{v}) = \sum_{j=1}^{s} v_{j,t}\, \psi_2(\mathbf{x}_t, \mathbf{c}_j) = \sum_{j=1}^{s} v_j o_{j,t}, \qquad t = 1, 2, ..., N \tag{3}$$

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 $\hat{y}_t$ as follows

$$\hat{y}_t = G(\mathbf{x}_t, \mathbf{c}, \mathbf{v}) = \sum_{j=1}^{s} v_{j,t} \frac{o_{j,t}}{\sum_{j=1}^{s} o_{j,t}} = \sum_{j=1}^{s} v_{j,t} \frac{\psi_2(x_t, c_j)}{\sum_{j=1}^{s} \psi_2(x_t, c_j)}, \qquad t = 1, 2, ..., N. \tag{4}$$

The network with one hidden layer and normalised output values $o_{j,t}$ is the fuzzy logic model or the soft RBF network (see Fig. 2(b)).

Basically, there are two learning schemes to adapt the weights $v_{j,t}$. The first one uses the linear estimation function of the column weight vector $\mathbf{v}$ as an optimal solution in a least squares sense. In vector notation this estimation function for output layer weights is

$$\mathbf{v}_t = (\mathbf{O}^{\mathrm{T}}\mathbf{O})^{-1}\mathbf{O}^{\mathrm{T}}\mathbf{y} \tag{5}$$

where $\mathbf{v}_t^{\mathrm{T}} = (v_{1,t}, v_{2,t}, ..., v_{s,t})$, $\mathbf{O}$ is a $(N \times s)$ matrix and $\mathbf{y}$ is the $(N \times 1)$ vector. The matrix $(\mathbf{O}^{\mathrm{T}}\mathbf{O})$ is of the dimension $(s \times s)$ easily inverted for very large number of data samples.

The second learning scheme uses the first-order gradient procedure. In our case, the subjects of learning are the weights $v_{j,t}$ only. These weights can be adapted by the error back-propagation algorithm. In this case, the weight update is particularly simple. If the estimated output for the single output neuron is $\hat{y}_t$, and the correct output should be y_t, then the error e_t is given by

$$e_t = y_t - \hat{y}_t$$

and the learning rule has the form

$$v_{j,t} \leftarrow v_{j,t} + \eta\, o_{j,t}\, e_t, \qquad j = 1, 2, ..., s; \quad t = 1, 2, N \tag{6}$$

where the term η, $\eta \in (0,1)$ is a constant called the learning rate parameter, $o_{j,t}$ is the normalised output signal from the hidden layer. Typically, the updating process is divided into epochs. Each epoch involves updating all the weights for all the examples.

Next, to improve the abstraction ability of soft RBF neural networks with architecture depicted in Figure 5, we replaced the standard Gaussian activation (membership) function of RBF neurons with functions based on the normal cloud concept [7].

Definition: Let *U* be the universe of discourse. *A* is a qualitative concept valued on *U*. The certainty degree $\mu_A(x)$ of a random sample *x* of *A* in *U* to the concept *A* is a random number with a stable tendency. Then the distribution of *x* on *U* is called a cloud model and *x* is called a cloud drop.

Cloud models are described by three numerical characteristics: 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. (9) has the form

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

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

4. Empirical Comparison

The RBF NN was trained using the variables and data sets as each ARCH-GARCH model above. The architecture (5-1-1) of the network, reported in the last column of Table 2, consists of three layers and seven neurons: five neurons in input layer, one in hidden and output layer.

In the granular RBF neural network framework, the non-linear forecasting function f(x) was estimated according to the expressions (9) with RB function $\psi_2(./.)$ given by equation (2). Granular RBF NN assumes that the noise level of the entropy is known. 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. The forecasting ability of particular networks was measured by the MSE criterion of ex post forecast periods (validation data set). The detailed computational algorithm for the forecast MSE values and the weight update rule for the granular network is shown in Appendix. The result of this application is shown in Table 1. A direct comparison between statistical (ARCH-GARCH) and neural network models shows that the statistical approach is better than the neural network competitor. The achieved ex post accuracy of RBF NN (RMSE = 0.00700), and for the statistical model ARCH-GARCH type with GED error distribution RMSE = 0.00106 (see [8]), but is still reasonable and acceptable for use in forecasting systems that routinely predict values of variables important in decision processes. Moreover, as we could see, the RBF NN has such attributes as computational efficiency, simplicity, and ease adjusting to changes in the process being forecast.

5. Conclusion

This paper has presented the granular RBF neural network based on Gaussian activation function modelled by cloud concept for solving approximation and prediction problems of real financial and economic processes. The neural network is suggested as an alternative to widely used statistical and econometric techniques in time series modelling and forecasting.

The power of the granular RBF NN is tested against some nonlinear high frequency data. A comparative analysis of two empirical studies is executed in order to evaluate its performance. The presented neural network, or soft computing approach, is applied on real data and time series with different models. It is able by using input-output data to find a relevant functional structure between the input and the output.

Table 1: Ex post forecast RMSEs for various granular RBF NNs (see text for details).

Number of RBF neurons (*s*)	K^*	RMSE
1	1.25 4.00	0.00719 0.00716
5	1.25 4.00	0.00756 0.00758
10	1.25 4.00	0.00720 0.00715

Acknowledgments. This paper/article/patent/functional sample/... has been elaborated in the framework of the IT4Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 supported by Operational Programme 'Research and Development for Innovations' funded by Structural Funds of the European Union and state budget of the Czech Republic and by the grant GACR P402/12/0070.

References

[1] MARCEK, M., PANCIKOVA, L., MARCEK, D. 2008. *Econometrics & Soft Computing*. Zilina: 2008, The University Press 271 p.

[2] MACIAROVA Z. 2009: Computing and Its Application in RBF Neural Network. with Cloud Activation Function. *Journal of Information, Control and Management Systems*, Faculty of Management Science and Informatics, University of Zilina, 7(1), 71-79.

[3] MACIAROVA Z. 2008: Neural network with Cloud Actication Function and Time Series in Managerial Forecasting Systems. *Journal of Information, Control and Management Systems*, Faculty of Management Science and Informatics, University of Zilina, 7(1), 93-100.

[4] ZADEH, L.A. 1979. Fuzzy sets and information granulation. Advances in Fuzzy Set Theory and Applications. M. Gupta, R. Ragade and R. Yager (Eds.), Amsterdam: North Holland Publishing Co., 3-18.

[5] ZADEH, L.A. 2008. Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logics. *Fuzzy Sets and Systems*, Vol. 19, 111-127.

[6] YAO, Y.Y. 2008. Granular Computing: Past, Present and Future, Proc. of the IEEE Int. Conf. on Granular Computing. Hangzhou 26th – 28th August, 80-85.

[7] LI, D., DU Y. 2008. Artificial Intelligence With Uncertainty. Boca Raton: 2008, Chapman & Hall/CRC, Taylor & Francis Group, 376 p.

[8] MARČEK, M.: *Statistical and RBF NN models: Providing forecast and risk assessment*. Ostrava:2009, *Ekonomická revue - Central European Review of Economic Issues* 12 (4), (175-182).

Summary

Článok prezentuje vývoj predikčných modelov založených na neurónových sieťach pre prognózovanie cien dlhopisov VUB banky a zhodnocuje sa ich predilčná presnosť s klasickými štatistickými modelmi typu ARCH-GARCH. Existuje len obmedzená štatistická teória alebo teória v počítačových vedách o navrhovaní architektúry a modelov RBF sietí pre špecifický finančný alebo ekonomický časový rad, pomocou ktorého by bolo možné študovať vnútornú dynamiku finančného procesu a identifikovať parametre takéhoto modelu. Aby mohla byť určená predikčná výkonnosť týchto prístupov, článok na konkrétnej aplikácii prezentuje učiace aspekty vývoja týchto modelov. Súčasne sa navrhuje nový prístup (metóda) pre identifikáciu funkcie procesu, ktorý tento časový rad odráža pomocou granulárnej neurónovej siete s Gaussovou aktivačnou funkciou a so zakomponovaním cloud konceptu (tzv. cloud computing). V komparatívnej štúdii je ukázané, že pomocou uvedeného postupu identifikácie finančných procesov je možné modelovať a predikovať vysokofrekvenčné finančné dáta s prijateľnou presnosťou a oveľa efektívnejšie než štatistickými metódami založenými na ARCH-GARCH metodológii.