International Journal of Computational Intelligence Systems, Vol.2, No. 4 (December, 2009), 353-364

Granular RBF NN Approach and Statistical Methods Applied to Modelling and Forecasting High Frequency Data

Dusan Marcek[1,2], **Milan Marcek**[3,4], **Jan Babel**[5]

[1] *Faculty of Management Science and Informatics*
University of Zilina, Univ. Street 1, Zilina, 010 26, Slovak Republic
E-mail: dusan.marcek@fri.uniza.sk

[2] *Institute of Computer Science, Silesian University,*
Bezruc Square 13, Opava, 746 01, Czech Republic
E-mail: dusan.marcek@fpf.slu.sk

[3] *MEDIS Nitra, Ltd., Pri Dobrotke 659/81, Nitra, 949 01,*
Slovak Republic
E-mail: marcek@medis.sk

[4] *Institute of Computer Science, Silesian University,*
Bezruc Square 13, Opava, 746 01, Czech Republic

[5] *Department of Macro & Micro Econimics, University of Zilina,*
Univ. Street 1, Zilina, 010 26, Slovak Republic
E-mail: jan.babel@fri.uniza.sk

Received: 14/01/09
Accepted: 12/05/09

Abstract

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

Keywords: Time series, classes of ARCH-GARCH models, volatility, forecasting, neural networks, cloud concept, forecast accuracy.

1. Introduction

Over the past ten years academics of computer science have developed new soft techniques based on latest information technologies such as soft, neural and granular computing to help predict future values of high frequency financial data which are observations on financial variables taken daily or at a finer time scale, and are often irregularly spaced over time (see Ref.1). At the same time, the field of financial econometrics has undergone various new developments, especially in finance models, stochastic volatility and software availability.

Volatility is an important factor in options trading. By volatility we mean the conditional standard deviation of asset prices. Volatility has many financial applications. For example volatility modelling provides a simple approach to calculating value at risk of a financial position in risk management. It also plays an important role in asset allocations under the mean-variance framework.

This paper discusses and compares the forecasts of volatility models which are derived from competing statistical and Radial Basic Function (RBF) neural network (NN) specifications. Our motivation for this comparative study lies in both the difficulty for constructing of appropriate statistical Autoregressive / Generalised Conditionally Heteroscedastic (ARCH-GARCH) models (so called hard computing) to forecast volatility even in ex post simulations and the recently emerging problem-solving methods that exploit tolerance for impression to achieve low solution costs (soft computing).

Recently, most developed statistical (econometric) models assume a nonlinear relationship among variables, for example the exponential and power GARCH models and exponential autoregressive models. These are model-driven approaches based on a specific type relation among the variables. On the other hand, neural networks and other soft computing techniques,are data driven models and nonparametric models. Unlike in classical statistical inference, the parameters are not predefined and their number depends on the used training data. Parameters that define the capacity of model are data-driven in such a way as to match the model capacity to the data complexity (see Ref. 2 for details). In this paper the relative forecasting and approximation performance of nonlinear statistical models ARCH-GARCH, EGARCH (Exponential GARCH), PGARCH (Power GARCH) and an ARMA (AutoRegressive Moving Average) model respectively are compared with granular NN models.

The paper is organized as follows. In the Section 2 we briefly describe the basic ARCH-GARCH model and its variants: EGARCH, PGARCH models. In the Section 3 we present the data, conduct some preliminary analysis of the time series and demonstrate the forecasting abilities of ARCH-GARCH modes of an application. In the Section 4 we introduce the architectures of RBF (Radial Basic Function) networks. The Section 5 includes an empirical comparison. The Section 6 briefly concludes.

2. Some ARCH-GARCH Models for Financial Data

ARCH-GARCH models are designed to capture certain characteristics that are commonly associated with financial time series. They are among others: fat tails, volatility clustering, persistence, mean-reversion and leverage effect. As far as fat tails, it is well know that the distribution of many high frequency financial time series usually have fatter tails than a normal distribution. A phenomenon of fatter tails is also called excess kurtosis. In addition, financial time series usually exhibit a characteristic known as volatility clustering in which large changes tend to follow large changes, and small changes tend to follow small changes. Volatility is often persistent, or has a long memory if the current level of volatility affects the future level for more time periods ahead. Although financial time series can exhibit excessive volatility from time to time, volatility will eventually settle down to a long run level. The leverage effect expresses the asymmetric impact of positive and negative changes in financial time series. It means that the negative shocks in price influence the volatility differently than the positive shocks at the same size. This effect appears as a form of negative correlation between the changes in prices and the changes in volatility.

The first model that provides a systematic framework for volatility modelling is the ARCH model was proposed by Engle (see Ref. 3). Bollerslev (see Ref. 4) proposed a useful extension of Engle‘s ARCH model known as the generalised ARCH (GARCH) model for time sequence $\{y_t\}$ in the following form

$$y_t = v_t \sqrt{h_t}$$
$$h_t = \alpha_0 + \sum_{i=1}^{m} \alpha_i y_{t-1}^2 + \sum_{j=1}^{s} \beta_j h_{t-j} \qquad (1)$$

where $\{y_t\}$ is a sequence of IID (Independent Identical Distribution) random variables with zero mean and unit variance. α_i a β_i are the ARCH and GARCH parameters, h_t represent the conditional variance of time series conditional on all the information to time $t-1$, the information set available at time I_{t-1}. In the literature several variants of basic GARCH model (1) has been derived. In the basic GARCH model (1) if only squared residuals ε_{t-i} enter the equation, the signs of the residuals or shocks have no effects on conditional volatility. However, a stylized fact of financial volatility is that bad news (negative shocks) tends to have a larger impact on volatility than good news (positive shocks). Nelson (see Ref. 5) proposed the following exponential GARCH model abbreviated as EGARCH to allow for leverage effects in the form

$$\log h_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i \frac{|\varepsilon_{t-i}| + \gamma_i \varepsilon_{t-i}}{\sigma_{t-i}} + \sum_{j=1}^{q} \beta_j h_{t-j} \quad (2)$$

Note if ε_{t-i} is positive or there is "good news", the total effect of ε_{t-i} is $(1+\gamma_i)\,\varepsilon_{t-i}$. However contrary to the "good news", i.e. if ε_{t-i} is negative or there is "bad news", the total effect of ε_{t-i} is $(1-\gamma_i)\,|\varepsilon_{t-i}|$. Bad news can have a larger impact on the volatility. Then the value of γ_i would be expected to be negative (see Ref. 1, p. 241).

The basic GARCH model can be extended to allow for leverage effects. This is performed by treating the basic GARCH model as a special case of the power GARCH (PGARCH) model proposed by Ding, Granger and Engle (see Ref. 6):

$$\sigma_t^d = \alpha_0 + \sum_{i=1}^{p} \alpha_i \left(|\varepsilon_{t-i}| + \gamma_i \varepsilon_{t-i}\right)^d + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^d \quad (3)$$

where d is a positive exponent, and γ_i denotes the coefficient of leverage effects (see Ref. 1, p. 243).

Finally the ARMA(p,q) model, often called Box Jenkins model is defined as follows:

$$y_t = c + \varepsilon_t + \sum_{i=1}^{p} \phi_i y_{t-i} \varepsilon_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \qquad (4)$$

where c is a constant, ε_t are the error terms that are generally assumed to be independent identically-distributed random variables sampled from a normal distribution with zero mean and constant variance $\varepsilon_t \sim \mathrm{N}(0,\sigma^2)$. ϕ and θ are parameters of the AR an MA parts (see Ref.7).

Another ARCH-GARCH models such as the ARCH-GARCH regression or ARCH-GARCH in-the-mean model can be found in Ref. 8, p.155-156.

3. An Application of ARCH-GARCH Models

We illustrate the ARCH-GARCH methodology by developing a forecasting models. The data is taken from the commercial VUB bank of the Slovak Republic and are available at http://www.vubam.sk/Default.aspx?CatID=40&fundId=4. The data is also listed at http://fria.fri.uniza.sk/files/data_VUB. The data consist of daily observations for the price time series of the bond fund of VUB (BPSVUB). The data was collected for the period May 7, 2004 to February 28, 2008 which provided of 954 observations (see Figure 1 and 4). To build a forecasting model, the sample period for analysis $r_1, \ldots, r_{900}$ was defined (denoted as training data set), i.e. the period over which the forecasting model was developed and the ex post forecast period (validation data set denoted) $r_{901}, \ldots, r_{954}$ as the time period from the first observation after the end of the sample period to the most recent observation. By using only the actual and forecast values within the ex post forecasting period only, the accuracy of the model can be calculated.

Input selection is crucial importance to the successful development of an ARCH-GARCH model. Potential inputs were chosen based on traditional statistical analysis: these included the raw BPSVUB and lags thereof. The relevant lag structure of potential inputs was analysed using traditional statistical tools, i.e. using the autocorrelation function (ACF), partial autocorrelation function (PACF) and the Akaike/Bayesian information criterion (AIC/BIC)(see Ref. 8): we looked to determine the maximum lag for which the PACF coefficient was statistically significant and the lag given the minimum AIC. According to these criterions the ARMA(5,0) model was specified as

$$r_t = \xi + \phi_1 r_{t-1} + \phi_2 r_{t-2} + \phi_3 r_{t-3} + \phi_4 r_{t-4} + \phi_5 r_{t-5} + \varepsilon_t \tag{5}$$

where $\xi, \phi_1, \phi_2, \ldots, \phi_5$ are unknown parameters of the model, ε_t is independent random variable drawn from stable probability distribution with mean zero and variance σ_ε^2.

As we mentioned early, high frequency financial data, like our BPSVUB, reflect a stylized fact of changing variance over time. An appropriate model that would account for conditional heteroscedasticity should be able to remove possible nonlinear pattern in the data. Various procedures are available to test an existence of ARCH or GARCH. A commonly used test is the LM (Lagrange Multiplier) test. The LM test assumes the null hypothesis $H_0: \alpha_1 = \alpha_2 = \ldots = \alpha_p = 0$ that there is no ARCH. The LM statistics has an asymptotic χ^2 distribution with p degrees of freedom under the null hypothesis. For calculating the LM statistics see for example Ref. 3 Eqs. (27) and (28). The LM test performed on the BPSVUB indicates presence of autoregressive conditional heteroscedasticity. For estimation the parameters of an ARCH or GARCH model the maximum likelihood procedure was used. The quantification of the model was performed by means of the R 2.6.0 software at http://cran.r-project.org has resulted into the following mean equation:

$$r_t = 0.0000748 + 0.06628 r_{t-1} + 0.09557 r_{t-2} +$$
$$+ 0.0528 r_{t-3} + 0.0528 r_{t-4} + 0.09795 r_{t-5} + e_t$$

and variance equation

$$h_t = 1.95810^{-8} + 0.1887 e_{t-1}^2 + 0.8075 h_{t-1} \tag{6}$$

where e_t are estimated residuals of ε_t from Eq. (5).

Finally, to test for nonlinear patterns in price bond time series, the fitted standardized residuals $\widehat{\varepsilon_t} = e_t/\sqrt{h}$ were examined by the BDS test.The BDS test (at dimensions N = 2, 3, and tolerance distances ε = 0.5, 1.0, 1.5, 2.0) finds no evidence of nonlinearity in standardized residuals of the BPSVUB. The BDS statistic is based on the null hypothesis that data in a time series is independently and identically distributed (idd) against an unspecified alternative (see Ref. 9).The BDS statistic is defined as

$$BDS_{m,N}(\varepsilon) = (N - m + 1)^{1/2}[C_{m,N}(\varepsilon) - (C_{1,N}(\varepsilon))^m] \tag{7}$$

where $C_{m,N}(\varepsilon)$ is a value of correlation integral or a number of clustered pairs lying within a particular tolerance distance ε at a spatial dimension m. The correlation integral $C_{m,N}(\varepsilon)$is given by

$$C_{m,N}(\varepsilon) = \sum_{t=1}^{N} \sum_{S=t-1}^{N} I_\varepsilon x_t^m, x_s^m)[2/N_m(N_m - 1)]$$

where $x_t^m = x_1, x_2, \ldots, x_{t+m-1}$ is m-dimensional vector of a scalar time series $\{x_t\}$ of length N and where $I_\varepsilon x_t^m, x_s^m = \|x_t^m - x_s^m\| = sup \mid x_{t-1} - x_{s+1} \mid < \varepsilon$. Thus the correlation integral measures fraction of pairs that lie within the tolerance distance ε for particular spatial dimension m. For more details see Ref. 8. The fitted vs. actual values are graphically displayed by means of the Eviews software (http://www.eviews.com) in Figure 1. The volatility was estimated by means of the R 2.6.0 software and is displayed in Figure 2.

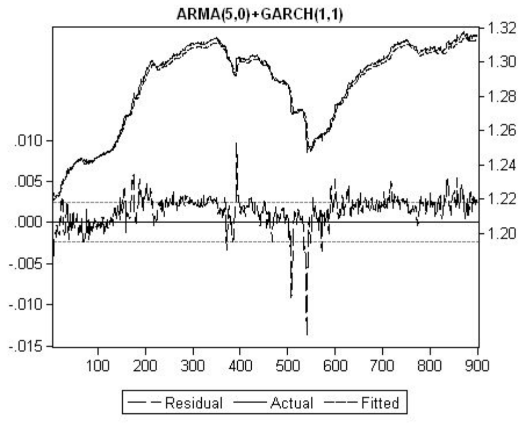


Fig. 1. Actual and fitted values of the VUB fund: ARMA(5,0)+GARCH(1,1) - model (6). Residuals are at the bottom. Actual time series represents the solid line, the fitted vales represents the dotted line.

Finally, for catching the leverage effect, the ARMA(5,0)+EGARCH(1,1) model was estimated. The coefficient for leverage effect γ from equation (2) is statistical significant and equals -0.2099535, and it is negative which means that ”bad news” have larger impact to volatility. If we comparing the estimated volatility in Figure 2 with the VUB fund in Figure 1, we can see that in period of depression the ”leverage effects” and the bad news cause the asymmetric jump in the volatility.

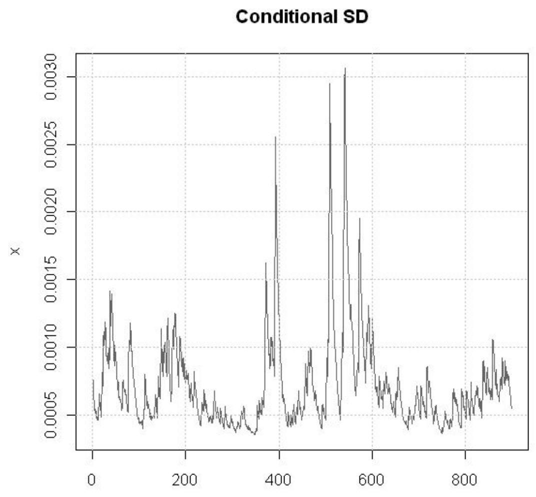
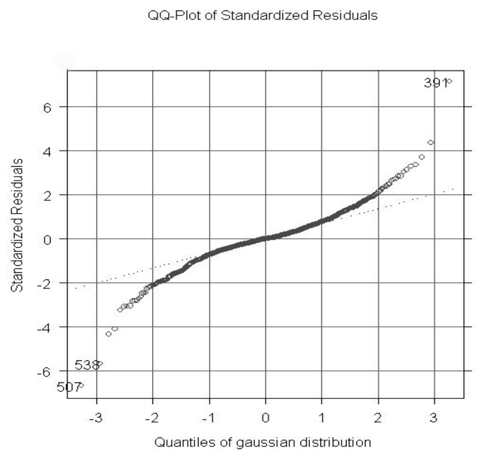


Fig. 2. The estimated volatility for ARMA(5,0) + GARCH(1,1) process - model (6).

In many cases, the basic GARCH model with normal Gaussian error distribution (1) provides a reasonably good model for analyzing financial time series and estimating conditional volatility. However, there are some aspects of the model which can be improved so that it can better capture the characteristics and dynamics of a particular time series. For this purpose the Quantile-Quantile (QQ) plots are used. For example, the R system (http://cran.r-project.org/) assist in performing residual analysis (computes the Gaussian, studentised and generalized residuals with generalized error distribution - GED). In Figure 3(a) the QQ-plot reveals that the normality assumption of the residuals may not be appropriate. According to Figures 3 a comparison of QQ-plots shows that GED distribution promise better goodness of fit. This is confirmed by AIC and BIC criterions and Likelihood function displayed in Table 1. The GED error distribution provides the best fit because AIC and BIC criterions are the smallest.

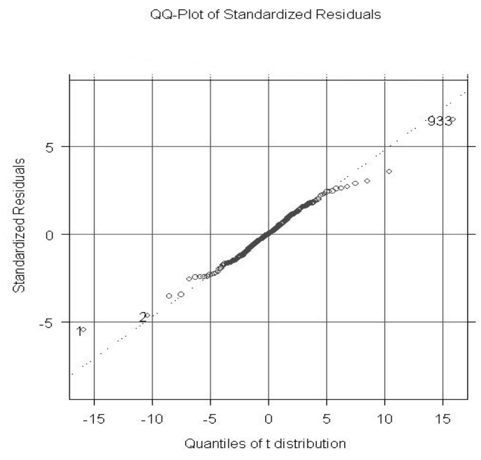
Table 1. AIC, BIC and likelihood function for various types error distribution (model (6)).

Model	model.n (Gaussian)	model.t (studentised)	model. GED
AIC criterion	-10576	-10778	-10792
BIC criterion	-10533	-10730	-10744
Likelihood funct.	5297	5399	5406



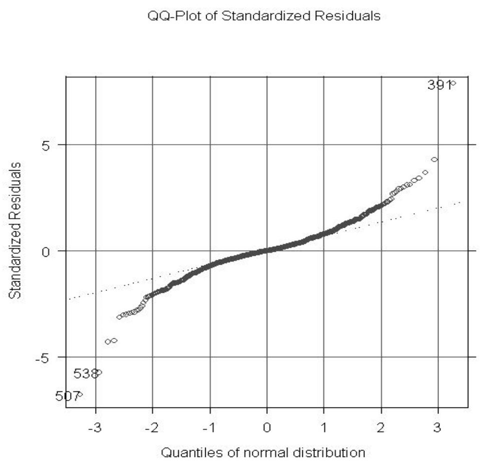
(a)

Fig. 3. QQ-plot of Gaussian standard residuals (a), studentised residuals (b) and generalised (GED) residuals (c).



(b)

Fig. 3. (Continued).



(c)

Fig. 3. (Continued).

As we mentioned above, the estimation of EGARCH and PGARCH models has showed the presence of leverage effects. The assumption of normal error distribution is also violated because the alternative error distributions provide better goodness of their fit. These findings indicate the chance of gaining better results in forecasting with using some of these models. Our suspicion was confirmed by computing the statistical summary measure of the model‘s forecast RMSE (Root Mean Square Error).

As we can see in Table 2 the smallest errors have just the GARCH with GED distribution.

After these findings we can make predictions for next 54 trading days using the model with the smallest RMSE, i.e. by the ARMA(5,0) + GARCH(1,1) model with GED error distribution. These predictions are calculated by means of the Eviews software (http://www.eviews.com) and shoved in Figure 4.

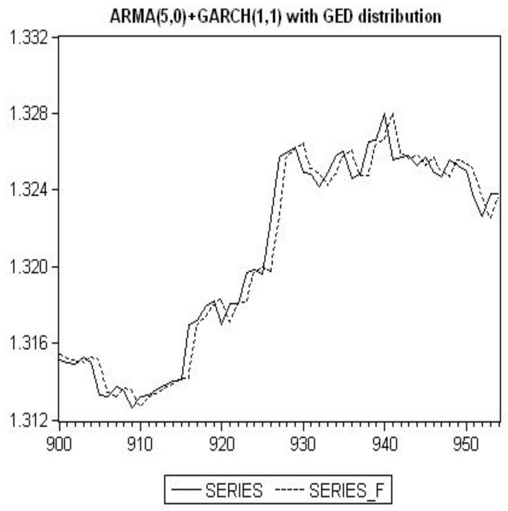


Fig. 4. Actual (solid) and forecast (dotted) values of the VUB fund.

Table 2. Ex post forecast RMSEs for various extensions of GARCH models.

Model
Distribution
	AR(5)+
	GARCH(1,1)
		AR(5)+
		EGARCH(1,1)
			AR(5)
			PGARCH(1,1)

Gaussian	0.003461	0.001066	0.001064
t-distribution	0.002345	0.001064	0.001063
GED-distribution	0.001056	0.001063	0.001062

4. An Alternative Approach

Over the past few years, new information technologies based on the fact that financial time series contain nonlinearities have been developed which document competitive performance of NN on a larger number of time series (see Ref. 10,11). 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 (see Ref. 12). We proposed the neural architecture according to Figure 5.

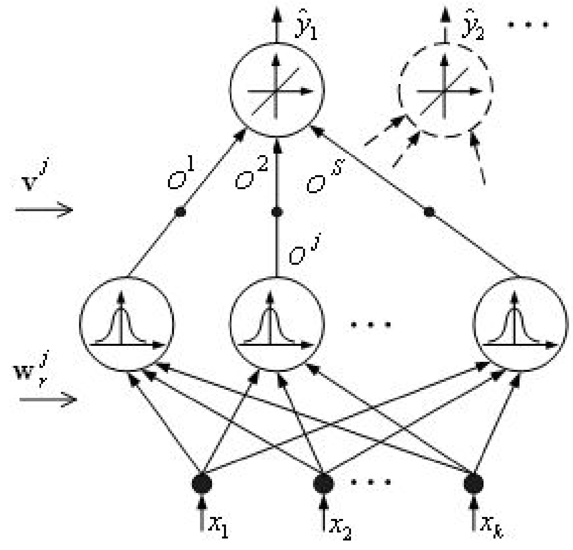


Fig. 5. RBF neural network architecture.

The structure of a neural network is defined by its architecture (processing units and their interconnections, activation functions, methods of learning and so on). In Figure 5 each circle or node represents the neuron. This neural network consists of an input layer with input vector $\boldsymbol{x}$ and an output layer with the output value $\widehat{y}_t$. 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

$$o_j = \psi_2\left(\left\|x - w_j\right\|\right) \tag{8}$$

where $\boldsymbol{x}$ is a k-dimensional neural input vector, 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 w_j represent the centres c_j of activation functions in the hidden layer. To find the weights 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 $c_j^{(t)}$, j = 1, 2, , s where s represents the number of chosen RBF neurons (clusters).

Step 2. Apply the new training vector

$$x^{(t)} = (x_1, x_2, \ldots, x_k)$$

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

$$c_j^{(t+1)} = c_j^{(t)} + \lambda\left(t\right)\left(x^{(t)} - c_j^{(t)}\right)$$

where $\lambda\left(t\right)$ is the learning coefficient selected as linearly decreasing function of t by $\lambda\left(t\right) = \lambda_0\left(t\right)$ $\left(1 - t/N\right)$ where $\lambda_0\left(t\right)$ 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 RBF function, the standard deviation, is estimated as K, $(K \geqslant 1)$ multiple of the mean value of quadratic distance among the the input vectors and their cluster centres. The value of K is regarded as the rate of overlapping in the distribution of input data (see Ref. 13). The above learning method based on the clustering algorithm is regarded as one of the granular method presenting the bottom-up granulation (see Ref. 14). Input vectors are combined into larger overlaping granules (clusters) described by clusters centres and the standard deviations.

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

$$\widehat{y} = \sum_{j=1}^{s} v_j o_j \tag{9}$$

where v_j are the trainable weights connecting the component of the output vector $\boldsymbol{o}$. 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

$$\widehat{y}_t = G(x_t, c, v) = \sum_{j=1}^{s} v_{j,t} \psi_2(x_t, c_j) = \sum_{j=1}^{s} v_j o_{j,t} \qquad t = 1, 2, \ldots, N \quad (10)$$

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 $(\|x - c_j)\|$ and compute the scalar values $o_{j,t}$ of the Gaussian function $\psi_2(x_t, c_j)$ that form the hidden layer output vector $\boldsymbol{o}_t$. Finally, the single linear output layer neuron computes the weighted sum of the Gaussian functions that form the output value of $\widehat{y}_t$.

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 $\widehat{y}_t$ as follows

$$\widehat{y}_t = G(x_t, c, v) = \sum_{j=1}^{s} v_{j,t} \frac{o_{j,t}}{\sum_{j=1}^{s} o_{j,t}} = \sum_{j=1}^{s} v_{j,t} \frac{\psi_2(x_t, c_j)}{\sum_{j=1}^{s} \psi_2(x_t, c_j)} \qquad t = 1, 2, \ldots, N \quad (11)$$

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

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

$$\mathrm{v}_t = \left(\mathrm{O}^{\mathrm{T}}\mathrm{O}\right)^{-1} \mathrm{O}^{\mathrm{T}} y \quad (12)$$

where $\mathrm{v}_{\mathrm{t}}^{\mathrm{T}} = (v_{1,t}, v_{2,t}, \ldots, v_{s,t})$, O is an $(N \times s)$ matrix and y is the $(N \times T)$ vector. The matrix $\left(\mathrm{O}^{\mathrm{T}}\mathrm{O}\right)$ 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 $\widehat{y}_t$, and the correct output should be y_t, then the error e_t is given by $e_t = y_t - \widehat{y}_t$ and the learning rule has the form

$$v_{j,t} \leftarrow v_{j,t} + \eta o_{j,t} e_t, \quad j = 1, 2, \ldots, s \qquad t = 1, 2, \ldots, N \quad (13)$$

where the term, $\eta \in (0,1)$ 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 (see Ref. 12, p. 113, Ref. 15, p.212).

Definition. *Let U be the universe of discourse. A is a qualitative concept valued on U. The certainty degree $\eta_A(x)$ 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. Then, in the case of soft RBF network, the Gaussian membership function $\psi_2(./.)$ in Eq. (11) has the form

$$\psi_2(x_t, c_j) = \exp\left[-(x_t - E(x_j))/2(En')^2\right] = \\ = \exp\left[-(x_t - c_j)/2(En')^2\right] \quad (14)$$

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

5. Empirical Comparison

The RBF NN was trained by using of the variables and data sets as each ARCH-GARCH model above. The network has been developed at the Faculty of Management Science and Informatics, University of Zilina. The architecture of the network consists of 5 neurons in input layer, 1, 5 or 10 RBF neurons in hidden layer and 1 neuron in output layer. In the granular RBF neural network framework, the non-linear forecasting function $f(x)$ was estimated according to the expressions (11) with RBF function $\psi_2(./.)$ given by Eq. (14). 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 the 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 (Mean Square Error) 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 A. The results of this application for various architectures of granular RBF networks are shown in Table 3.

Table 3. Ex post forecast RMSEs for various granular RBF NNs (see text for details).

Number of RBF neurons	K	RMSE
1	1.25	0.00719
	4.0	0.00716
5	1.25	0.00756
	4.0	0.00758
10	1.25	0.00720
	4.0	0.00715

A direct comparison between statistical (ARCH-GARCH) and neural network models shows that the statistical approach is better than the neural network competitor(see the last row of Table 2 and first row of Table 3). The achieved ex post accuracy of RBF NN (RMSE = 0.00719), but is still reasonable and acceptable for use in forecasting systems that routinely predict values of variables important in managerial 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. ARCH-GARCH models require more costs of development, installation and operation in a management system, management comprehension and cooperetion, and often a lot of computational time.

In Ref. 16, the approximation ability of sales time series $\{y_t\}_{t=1}^{724}$ (the 724 daily sales for Hansa Flex company, 2004 - 2005) was analysed by various types (classic, soft, granular) of RBF NNs. The statistical specification of the model resulted into the following two equations:

$$y_t = \phi_1 y_{t-7} + \varepsilon_t \quad (15)$$

or

$$y_t - y_{t-7} = \theta_1 \varepsilon_{t-7} + \varepsilon_t \quad (16)$$

In the classic RBF NN framework the non-linear function $f(x)$ was estimated according to the expressions (8) and (9). In the case of fuzzy logic RBF NN, the non-linear input - output approximation function was estimated according to the expression (11). The approximation results measured by MSE were calculated analogously as in the case of ARCH-GARCH model. Their values of various RBF′s networks related to the different number of clusters are shown in Table 4. The detailed computational algorithm for calculating the MSE values for approximation is shown in Refs. 15 and 16. Comparing both approaches, i.e. models based on the statistical methodology (the MSE for model expressed by Eq. (15) is 0.779 and by Eq. (16) is 0.7466 respectively), and models based on RBF network approaches we see, that models based on RBF networks, are better approximation models, because the estimated values are close to the actual values. As shown in Table 4, models that generate the "best" MSE′s, are soft RBF networks.

Table 4. Approximation results of various RBF's networks related to the different number of clusters (RBF neurons).

Neural network architecture:	Gaussian (classic) RBF network	Gaussian with normalised outputs (soft RBF network)	Gaussian soft RBF (with normal cloud concept - granular network)
Number of RBF neurons		MSE	
	RBF network representations for model (15): $y_t = \theta_1 y_{t-7} + \varepsilon_t$		
3	1.439	0.698	0.729
5	0.729	0.693	0.716
10	0.687	0.675	0.678
15	0.697	0.681	0.678
	RBF network representations for model (16): $y_t - y_{t-7} = \theta_1 \varepsilon_{t-7} + \varepsilon_t$		
3	0.783	0.646	0.647
5	0.810	0.632	0.630
10	0.607	0.571	0.571
15	0.582	0.563	0.563

6. 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.

The importance of having good intelligent forecasting tools for time series is ever more important with increasing number of data when more effort must be devoted to development of efficient data handling and management procedures. The proposed methodology is believed to be helpful in future research and its applications.

Acknowledgement

This work was partially supported by the Agency of the Slovak Ministry of Education for the Structural Funds of the EU, under project ITMS: 26220120007, by the grant GACR 402/08/0022 and project No. CZ.1.07/2.3.00/09.0197.

Appendix A The algorithm for updating weights in the granular RBF network and for calculating statistical forecast summary measures (MSE).

```
function WEIGHT_UPDATE_and MSE(granular_RBF_network, examples, $\eta$, $s$, $c_j$, $\sigma_j$, $He_j$)
returns a network with the MSE value
    inputs: granular_RBF_network , MSE a Gaussian soft RBF network
    with normal cloud concept
    $N_A$ number of training data set
    $N_E$ number of validation data set
    examplesA, a set of $N_A$ input/output observed data pairs: $x$, $y$ (training data set)
    examplesB, a set of $N_B$ input/output observed data pairs: $x$, $y$ (validation data set)
    $\eta$, the learning rate
    $s$, the number of clusters (RBF neurons)
    $c_j$, the centre of the $j$-th cluster, $j$ =1, 2, …, $s$
    $\sigma_j$, the standard deviation $j$-th cluster, $j$ =1, 2, …, $s$
    $He_j$, the hyper entropy
    - Initialize weights: $v_j$,$j$ = 1, 2, ..., $s$ leading to the output neuron.
    - Initialize the learning rate: $\eta$.
    - Initialize the input values: $s, c_j, \sigma_j, He_j$ (see text for details)
        repeat
          MSE $\leftarrow$ 0
              for each example x in examplesA do
              /* Generate normally distributed random numbers $He'_j$
              with the means $\sigma_j$ and standard deviations $He_j$ */
                  $He'_j \leftarrow$ RUN-NORMAL-RANDOM-GENERATOR($\sigma_j, He_j$), $j = 1,2,\ldots,s$
                  /* Calculate the outputs from the RBF neurons */
                  $o_j \leftarrow \psi_2(x, c_j)$, $j = 1,2,\ldots,s$
                  /* where $\psi_2$ is the Gaussian function: $\psi_2 = \exp\left(-\left(x - c_j/He'_j\right)\right)$ */
                  /* Calculate the normalized outputs $o_j^{(N)}$ */
                  $o_j^{(N)} \leftarrow o_j / \sum_{j=1}^{s} o_j$, $j = 1,2,\ldots,s$
                  /* Calculate the output $\widehat{y}$ from the output neuron */
                  $\widehat{y} \leftarrow \sum_{j=1}^{s} v_j o_j^{(N)}$
                  /* Calculate the output layer neuron′s error $e$ */
                  $e \leftarrow y - \widehat{y}$
                  /* Update the output layer weight $v_j$ */
                  $v_j \leftarrow v_j + o_j^{(N)} e$, $j = 1,2,\ldots,s$
              end
                  /* Calculate the mean square error */
              for each example $x, y$ in examplesB do
                  MSE = MSE + $\left[y - v_j \sum_{j=1}^{s} \exp\left(-\left(x - c_j\right) * 2He_j\right)\right]^2$
              end
          MSE = MSE/N$_E$
        until MSE reaches minimum
        return MSE, network
```

References

1. E. Zivot, J. Wang, *Modeling Financial Time Series with* S-PLUS®, (Springer Verlag, NY, 2005).
2. V. Kecman, V., *Learning and soft computing: support vector machines, neural networks, and fuzzy logic*, (Massachusetts: The MIT Press, 2001).
3. R. F. Engle, Auto regressive Conditional Heteroscedasticity with Estimates of the Variance of United Kindom Inflation, *Econometrica* 50 (1982) 987-1007.
4. D. Bollerslev, Generalized Autoregressive Conditional Heteroscedasticity, *Journal of Econometrics* 31 (1986) 307-327.
5. D. B. Nelson, Conditional Heteroskedasticity in Asset Returns: a New Approach, *Econometrica* 59 (2) (1991) 347-370.
6. Z. Ding, C. W. Granger and R. F. Engle, A Long Memory Property of Stock Market Returns and a New Model, *Journal of Empirical Finance*, 1, (1993) 83-106.
7. G. Box, and G. Jenkins, *Time series analysis: Forecasting and control* (San Francisco: Holden-Day, 1970).
8. D. Marcek, and M. Marcek, *Time Series Analysis, Modelling and Forecasting with Applications in Economics* (The University Press, Zilina, 2001).
9. W. A. Brock, W. D. Dechert, J. A. Scheinkman, and B. LeBaron, *A Test for Independence Based on the Correlation Dimension* (Econometric Reviews, 15, 1996).
10. K. P. Liao and R. Fildes, *The accuracy of a procedural approach to specifying feedforward neural networks for forecasting* (Computers & Operations Research, 32, 2151-2169, 2005).
11. G. P. Zhang and M. Qi, *Neural network forecasting for seasonal and trend time series* (European Journal Of Operational Research, 160, 501-514, 2005).
12. D. Li and Y. Du, *Artificial intelligence with uncertainty* (Boca Raton: Chapman & Hall/CRC, Taylor & Francis Group, 2008).
13. CH. M. Bischop, *Neural Networks for Pattern recognition* (Oxford University Press, New York, 1995).
14. Y. Y. Yao, *Granular Computing for Data Mining* (Kissimmee: Congres of Data Mining, Intrusion Detection, Information Assurance and Data Network Security 2006. Vol. 6241. 16th - 17th April 2006, pp. 624105.1-624105.12).
15. M. Marcek, L. Pancikova and D. Marcek, *Econometrics & Soft Computing* (The University Press, Zilina, 2008).
16. M. Marcek and D. Marcek, *Granular RBF Neural Network Implementation of Fuzzy Systems: Application to Time Series Modelling*, J. of Multi-Valued Logic & Soft Computing, 14 (2008) 400-414.