PREDICTION OF HIGH-FREQUENCY DATA: APPLICATION TO EXCHANGE RATES TIME SERIES

Milan Marček
[1]Medis, spol. s r.o., Pri Dobrotke, 659/81, 948 01 Nitra,
Silesian University, Institute of computer Science, Opava, Czech Republic

Dušan Marček
University of Žilina, Faculty of Managemet Science and Informatics, Žilina, Slovak Republic

Abstract

This paper considers the use of ANN methodology for parameters estimation of the autoregressive conditional heteroscedastic (ARCH) processes. The paper provides heuristic approach of ARCH processes modelling. This approach is often employed to estimate the values of financial variables as rates of return, exchange rates, means and variances of inflation, stock market returns and price indexes and also to predict their variances.

Keywords

Ekonometrics, forecasting, RF neural network economic hypothesis ARCH/GARC models.

1 INTRODUCTION

The usual assumptions for the linear regression model are that the noise or disturbance terms ε_t have mean zero $[E(\varepsilon_t) = 0]$, constant but unknown variance $\sigma_\varepsilon^2 = V(\varepsilon_t)$, and that ε_t's are uncorrelated. Sometimes these assumptions are unreasonable. In this paper we will briefly consider the case when the symmetric variance - covariance matrix V of ε_t in Ordinary Least Squares (OLS) procedure have unequal diagonal elements, that is, the errors ε_t do not have equal (nonconstant) variances (heteroscedasticity), while some of the off-diagonal elements of V are zero. Heteroscedasticity can be caused by incorrect specification or by the use of the wrong functional form. To remove the heteroscedasticity it is necessary either to introduce or to eliminate some variables or to introduce some transformation so that the variance errors will be uniform across all observations.

Financial variables as rates of return, exchange rates, inflation, stock market returns, price indexes and other economic variable of generally high frequency, are likely to originate from low complexity chaos and respond not only to the mean, but also to the variance and to the higher moments of random variables. It is widely know that high frequency economic, especially financial data change their variances over time. Detection of implausible variance behaviour in such time series provides important information which can improve the forecasting ability over short time period. The autoregressive conditional heteroscedastic process is based on such assumption that the recent past gives information about the one-period forecast variance. It is assumed that the variance of a random variable forecast y_t is depend on the value of the conditioning variable y_{t-1} that is $V(y_t | y_{t-1})$. ARCH models are going widely applied to series, where hourly, daily or weekly observations are of interest.

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. At the same time, the field of financial econometrics has undergone various new developments, especially in finance models, stochastic volatility, and software availability.

This paper discusses and compares the forecasts from 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).

The paper is organized as follows. In Section 2 we briefly describe the basic ARCH-GARCH model. In 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 Section 4 we introduce the architecture of fuzzy logic RBF neural network for time series forecasting. Section 5 compares both approaches and briefly concludes.

2 ARCH MODELS

As we mentioned above, the basic idea of ARCH models is to incorporate in standard regression or econometric model an independent variable, which evaluated or predicts the variance. The sample initial model introduced by Engle [4] is

$$y_t = \varepsilon_t \sqrt{h_t}$$
$$h_t = a_0 + a_1 y_{t-1}^2$$

or in terms of ψ_t the information set avaliable at time *t*, using conditional densities

$$y_t | \psi_{t-1} \sim \mathrm{N}(0, h_t) \tag{1}$$

where ε_t is random error component (white noise) with $V(\varepsilon_t) = \sigma^2$ and normally distributed, a_0, a_1 are unknown parameters to be estimated and h_t is the variance function.

Bollerslev [2] 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

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

Model (2) forms the underlying basis for analysis of many financial processes. In [6] the models of dependence of exchange rate changes for four currencies (Deutsche Mark, Swiss Franc, French Franc, Japanise Yen) against US dollar are described by using six order of the ARCH model, represented in the following form

$$\Delta \log S_t = \varepsilon_t \left[\hat{a}_0 + \sum_{\tau\tau=1}^{6} \hat{a}_t (\Delta \log S_{t-\tau})^2 \right]^{\frac{1}{2}}, \ t = 1,2, \ldots, N$$

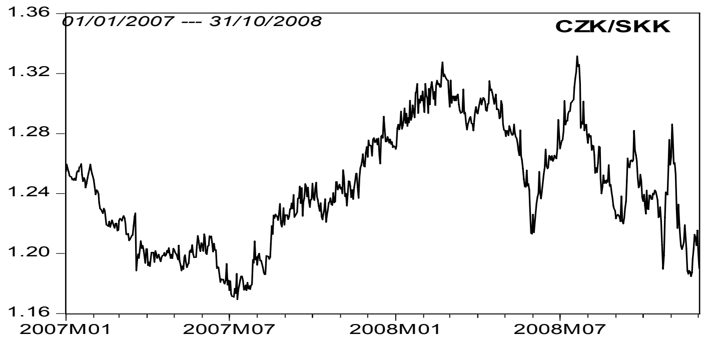
where $\Delta \log S_t = y_t$ is the weekly end rate of change of the log exchange rate by taking the first logarithmic difference, N is sample size.

ARCH-GARCH models are also 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. The phenomena 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. Another variants ARCH-GARCH models can be found in [9], [10], [7].

3 AN APPLICATIONOF ARCH-GARCH MODELS

We illustrate the ARCH-GARCH methodology on the developing a forecast model for the currency SKK/CZK (Slovak crown/Czech crown) during the period from January 1, 2007 to December 2, 2008. The length of the time series is 702 observations. The data are available at http://www.oamda.com. To build a forecast model the sample period (training data set denoted A) for analysis $r_1, ..., r_{670}$ was defined, and the ex post forecast period (validation data set denoted E) $r_{671}, ..., r_{702}$ The time plot of the data are shown in Fig. 1.

Figure 1: The evolution of cross rate CZK/SKK using the R system software [12]



We assume that the generating mechanism is probabilistic. The main purpose of time series analysis is to understand the underlying mechanism that generates the observed data and, in turn to forecast the futures values. In order to capture variance-nonlinearity an ARMA-GARCH model of changes rates was estimated. Te specification of the estimated model (2) with Gaussian error distribution resulted into AR(7)+GARCH(1,1) process with mean equation [1].

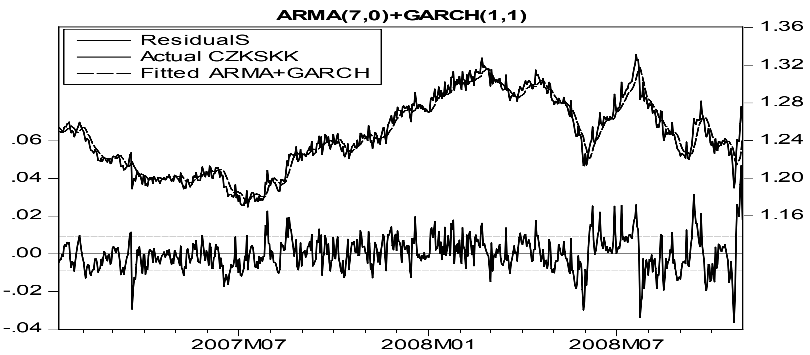
$$r_t = 0.0012 + 0.7269 r_{t-1} + 0.0929 r_{t-2} + 0.0843 r_{t-3} + 0.0821 r_{t-4} - 0.0651 r_{t-5} +$$
$$+ 0.05591 r_{t-6} + 0.0218 r_{t-7} + \varepsilon_t$$

and the variance equation (3)

$$h_t = 0.0000169 + 0.3049 \varepsilon_{t-1}^2 + 0.3616 h_{t-1}$$

The fitted model vs. the original dataset drawn by Eviews software [11] is shown in Figure 2.

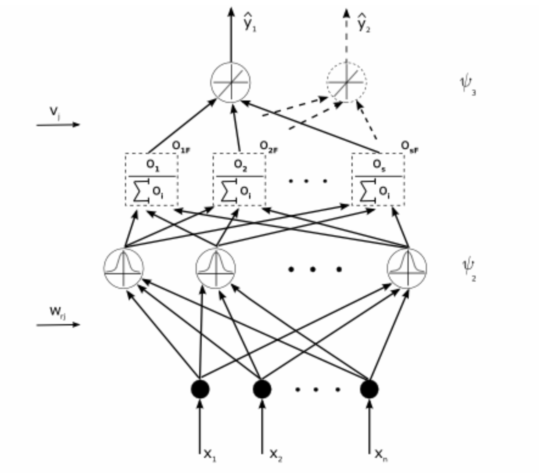
Figure 2: The estimated ARMA(7,0)+GARCH(1,1) model against the original datasets on whole range. (Residuals are at the bottom)



4 Soaft RBF Neural tworApproach

In this section we show an approach of function estimation for time series modelled by means a granular fuzzy logic RBF neural network based on Gaussian activation [5]. We proposed the neural architecture according to Figure 3.

Figure 3: Soft (fuzzy logic) NN



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 3 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 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

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

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 in the hidden layer. To find the weights $\mathbf{w}_j$ or centres of activation functions we used the adaptive (learning) version of K-means clustering algorithm for s clusters. The second parameter of the RB function, the standard deviation is estimated as K, (K $\geq$ 1) multiple 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. The RBF network computes the output data set as

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

where 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

$$\hat{y}_t = G(\mathbf{x}_t, \mathbf{c}, \mathbf{v}) = \sum_{j=1}^{s} v_{j,t} \frac{o_{j,t}}{\sum_{j=1}^{s} o_{j,t}} = \sum_{j=1}^{s} v_{j,t} \frac{\psi_2(x_t, c_j)}{\sum_{j=1}^{s} \psi_2(x_t, c_j)}, \quad t \in \mathrm{A} \tag{6}$$

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

To adapt the weights of $v_{j,t}$ we use 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

$$v_{j,t} \leftarrow v_{j,t} + \eta\, o_{j,t}\, e_t, \quad j = 1, 2, ..., s; \qquad t \in \mathrm{A} \tag{7}$$

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.

5 Empirical Comparison

The RBF NN was trained using the variables and data sets as each ARCH-GARCH model above. The architecture (7-2-1) of the network, reported in the last column of Table 1, consists of three layers and seven neurons: five neurons in input layer, one in hidden and output layer.

Model Distribution	AR(7)+ GARCH(1,1)	Soft RBF NN Topology (7-2-1)
Gaussian	0.01855	0.0403 η = 0.03, K= 2.6 $\lambda_0(t)$ = 0.02

Table 1: Ex post forecast RMSEs for ARCH-GARCH model and soft RBF NN. See text for details

In the fuzzy logic RBF NN, the non-linear forecasting function $f(\mathbf{x})$ was estimated according to the expressions (6) with Gaussian function $\psi_2(\mathbf{x}_t, \mathbf{c}_j)$. The forecasting ability of particular networks was measured by the RMSE (Root Mean Square Error) criterion of ex post forecast periods (validation data set). The detailed computational algorithm for the forecast RMSE values and the weight update rule for the network is shown [8]. 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 = 0.0403), 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 soft RBF NN has such attributes as computational efficiency, simplicity, and ease adjusting to changes in the process being forecast.

Acknowledgements

This work was supported by Czech grant foundation under the grant GACR No.: 402/08/0022

References

[1] BABEL, J., MECIAROVA, Z., PANCIKOVA, L.: *Statistical and Soft Computing Methods in Cross Rates Modeling*. Proceedings of seveth IC on SC Applied in Computer and Economic Environments.ICSC 2009, European Polytechnical Institute Kunovice, January 23, 2009, Czech Republic

[2] BOLLERSLEV, D.: *Generalized Autoregressive Conditional Heteroscedasticity*. Journal of Econometrics 31 (1986) 307-327

[3] DING, Z., GRANGER, C.W., and ENGLE, R F.: *A Long Memory Property of Stock Market Returns and a New Mode.* Journal of Empirical Finance, 1, (1993) 83-106

[4] ENGLE, R.F.: *Auto regressive Conditional Heteroscedas-ticity with Estimates of the Variance of United Kindom Inflation.* Econometrica 50 (1982) 987-1007

[5] KECMAN, V.: *Learning and soft computing: support vector machines, neural networks, and*

fuzzy logic. Massachusetts: The MIT Press, 2001

[6] KOCENDA, E.: *A Test for idd Residuals Based on Integrating over the Correlation Integral.* CERGE - EI, Working Paper 101, 1996

[6] LI, D., and DU, Y.: Artificial intelligence with uncertainty. Boca Raton: Chapman & Hall/CRC, Taylor & Francis Group, 2008

[7] MARCEK, D., and MARCEK, M.: *Time Series Analysis, Modelling and Forecasting with Applications in Economics.* The University Press, Zilina, 2001

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

[9] NELSON, D.B.: *Conditional Heteroskedasticity in Asset Returns: a New Approach.* Econometrica 59 (2) (1991) 347-370

[10] ZIVOT, E., WANG, J. (2005): *Modeling Financial Time Series with S-PLUS*®, Springer Verlag, 2005

[11] http://www.eviews.com

[12] http://www.forecasters.org/ijf

Milan Marček
milan.marcek@fri.uniza.sk

Dušan Marček
dusan.marcek@fri.uniza.sk