# Financial Time Series Modelling with Hybrid Model Based on Customized RBF Neural Network Combined With Genetic Algorithm

*Lukas FALAT*[1], *Dusan MARCEK*[2]

[1]Department of Macro & Microeconomics, Faculty of Management Science and Informatics, University of Zilina, Univerzitna 8215/1, Zilina, Slovakia
[2]Department of Applied Informatics, Faculty of Economics, VSB–Technical University of Ostrava, Sokolska 33, 701 21 Ostrava, Czech Republic

lukas.falat@fri.uniza.sk, dusan.marcek@vsb.cz

***Abstract.*** *In this paper, authors apply feed-forward artificial neural network (ANN) of RBF type into the process of modelling and forecasting the future value of USD/CAD time series. Authors test the customized version of the RBF and add the evolutionary approach into it. They also combine the standard algorithm for adapting weights in neural network with an unsupervised clustering algorithm called K-means. Finally, authors suggest the new hybrid model as a combination of a standard ANN and a moving average for error modeling that is used to enhance the outputs of the network using the error part of the original RBF. Using high-frequency data, they examine the ability to forecast exchange rate values for the horizon of one day. To determine the forecasting efficiency, authors perform the comparative out-of-sample analysis of the suggested hybrid model with statistical models and the standard neural network.*

## Keywords

***Artificial neural network, generic algorithm, hybrid model, RBF, time series, USD/CAD.***

## 1. Introduction

Predicting time series using statistical analysis started in the 60s years of 20th century. First statistical models were based on the theory of exponential smoothing originally published in [1], [2] and [3]. Another breakthrough came with publishing a study from Box & Jenkins [4]. In this study, they integrated all the knowledge including autoregressive and moving average models into one book. From that time, the ARIMA models have been very popular in time series modelling for a long time as O'Donovan [5] showed that these models provide better results than other models used in that time. However, in 1982 Engle [6] showed that using ARIMA models in financial series modelling is not always correct as these series usually have conditional variance instead of constant. Therefore, he suggested ARCH (autoregressive conditional variance) models for financial modeling.

One of the reason computers started to apply in time series modeling was the study of Bollershev [7] where he proved the existence of nonlinearity in financial data. The first techniques of machine learning applied into time series forecasting were artificial neural networks (ANN). As ANN was a universal approximator, it was believed that these models could perform tasks like pattern recognition, classification or predictions [8], [9], [10]. Today, according to some studies [11] ANNs are the models having the biggest potential in predicting financial time series. The reason for the attractiveness of ANNs for financial prediction can be found in works of Hill et al. [12], where authors showed that ANNs works best in connection with high-frequential financial data.

While in the first application of ANNs into financial forecasting, perceptron network, the simplest feedforward neural network, was used [13], nowadays it is mainly RBF network that is being used for this as they showed to be better approximators than the perceptron networks [14].

We decided to apply our neural networks models into the market of exchange rates. Forex, which is short for the foreign exchange market, is one of the world's largest and most liquid financial markets in the world. Therefore, it is no surprise that the exchange rates forecasting has attracted the attention of many financial researchers and analysts for a long time. Very common approach investors use for trading is technical analysis (TA). "Technical analysis can be seen as a collection of algorithms and mechanical rules which attempt to aid investors in forecasting future market movements, using only historic data" [15].

In this paper, forecasts of USD/CAD exchange rate through customized RBFGA neural network with Moving Average errors (RBFGA-SMA) was performed The standard RBF model will be extended by using Moving Average for modeling the errors of RBF network. Additionally, the hybridized version of ANN will be used; we will combine the standard ANN with EC technique called genetic algorithms that will be used in the process of finding optimal parameters of the neural network. According to some scientists [16], the use of technical analysis tools can lead to the efficient profitability on the market, we decided to combine our customized RBF network with a tool of the technical analysis. Investors use a large number of technical trading tools so as to make the decision-making process easier. In the process of searching an optimal tool of TA for hybridization with RBF, we were inspired by ARIMA models where the MA part is correcting the error of the AR model. Therefore, for our experiments one of the simplest tools of TA - Moving Averages was chosen. Just like many other tools of technical analysis, Moving Averages are also based on analyzing historical data using statistics. From a forecasting point of view, it is a type of finite impulse response filter used to predict the next value in the series by creating a series of averages of different subsets of the full data set. The simple moving average was used to model the error part of the RBF network as there was a suspicion it could enhance the prediction outputs of the model.

Our machine learning application to exchange rates forecasting is novel in two ways – we use the standard neural network hybridized with simple moving averages to form a whole new hybrid model for forecasting. Nowadays, there are lots of hybrid models of neural networks. Hassan et al. [17] tested a fusion model of Hidden Markov Models (HMM), ANN and GA for stock market forecasting. Thinyane and Millin [15] use the intelligent hybrid system composed of GA and ANN to enhance the decision-making process in the field of currency trading. Sterba and Hilovska [18] combine ANN and statistical ARIMA models to create a hybrid model. Other studies of hybrid models include, for example, those by Zhuang and Chan [19], Chikhi et al. [20], Choudhry and Garg [21] or Marcek and Falat [22]. However, none of the mentioned hybrid models does not focus on a combination of ANN with TA.

Moreover, we also use other than just the standard algorithm for training parameters of the neural network in order to achieve the maximal prediction accuracy of our suggested model.

## 2. Models and Methods

### 2.1. Box-Jenkins and GARCH Models

For more than 20 years Box-Jenkins ARMA models have been widely used for time series modelling. The models published in [4] are autoregressive models (AR) and moving average (MA) models. Let $y_t$ be a stationary time series that is the realization of a stochastic process. General formula of ARMA(p,q) model is expressed as follows (for details, see [4]):

$$y_t = \xi + \sum_{i=1}^{p} \phi_i \, y_{t-i} + \varepsilon_t - \sum_{j=1}^{q} \theta_i \varepsilon_{t-j}. \qquad (1)$$

The weakness of ARMA models is the inability to model non-constant variance. As this type of variance is very common in currency pairs, constant volatility is not able to capture some of the basic properties of heteroscedastic volatility present in financial time series such as stochasticity of volatility, volatility clustering, mean reversion and existence of fat tails. In [6] Engle suggested the solution by creating so-called ARCH (Autoregressive Conditional Heteroscedastic) models which assume heteroscedastic variance of $\varepsilon_t$.

### 2.2. Feed-Forward Neural Network

The model of artificial neural network based on human neural system is an universal functional black-box approximator of non-linear type [23], [24], [25] which are especially helpful in modelling non-linear processes having a priori unknown functional relations or this system of relations is very complex to describe [26].

In [23] and [24] it has been showed that a neural network can approximate any continuous function into any demanded accuracy. Moreover, artificial neural network can generalize. After learning on a training set data, the network is very often able to produce good outputs on unknown inputs.

Let $F$ be a function defined as:

$$F : x_t \in R^k \to y_t \in R^1, \qquad (2)$$

is a representation assigning one value $y_t$ to k-dimensional input a given time period $t$. Let $G$ be is a restriction of $F$ defined as:

$$G(x_t, w_t) : x_t \in R^k_{train} \to y_t \in R^1_{train}, \qquad (3)$$

where $R_{train}$ is a complement of $R_{val}$ to $R$. Then, artificial neural network is a mathematical model defined by finding the values $w_t$ so that the function in Eq. (1) would be minimal:

$$E(w_t) = \sum_{x_t y_t \in R^k_{train}} (G(x_t, w_t) - y_t). \qquad (4)$$

when $E$ minimal, one can say $G(x_t, w_t)$ is adapted to approximate the function $F$.

Multilayer perceptron (MLP) is the type of feed-forward neural network usually having three layers. Input layer is composed of the input vector; the output layer is represented by just one neuron and contains the network output. Usually, there is also one or more hidden layers between inputs and output. In most cases, one hidden layer is sufficient since according to Cybenko theorem [27] the network with one hidden layer is able to approximate any continuous function. Layers are interconnected via synapses (also called weights), which represent parameters of the neural network model.

## 2.3. Radial Basis Neural Network

Radial Basis Function (RBF) ANN is the upgrade of MLP network. The name of this type of neural network comes from the name of its activation function. Generally, a radial basis function (RBF) is real-valued functions whose values depend only on the distance from the origin or from some other point $c$, called a center:

$$\varphi(x, c) = \varphi(\|x - c\|). \tag{5}$$

Any function $\phi$ that satisfies the property in Eq. (7) is a radial function. The norm is usually Euclidean distance.

Hence, the biggest difference between MLP and RBF is in using the different function for activating hidden neurons. RBF neural network uses radial basis function of Gaussian type instead of the sigmoid function for activating neurons in the hidden layer. This function is defined for $j^{th}$ hidden neuron as:

$$\psi_1\left(u^j\right) = e^{\frac{-u^j}{2\sigma_j^2}} = e^{\frac{-\left\|x - w^j\right\|^2}{2\sigma_j^2}}, j = 1, 2, ...s, \tag{6}$$

where $\sigma_j^2$ is the variance of $j^{th}$ neuron. The network output for RBF neural network is then counted as follows:

$$y = \psi_2\left(\sum_j v_j * \psi_1(\|x - w\|)\right) = \sum_{j=1}^{s} v_j * e^{\frac{-\left\|x - w^j\right\|^2}{2\sigma_j^2}}. \tag{7}$$

## 2.4. Back-Propagation Algorithm

The most popular method for learning in multilayer networks is called back-propagation (BP). It was first invented in 1969 by Bryson and Ho [28], but was largely ignored until the mid-1980s. BP is a multi-stage dynamic system optimization method mainly used for adapting parameters of feed-forward neural networks. This algorithm, which is based on gradient descent and is a generalization of the delta rule, is a supervised learning method. Hence, the training set of desired outputs, according to which the adaptation is performed, is required. The assumption of using back-propagation is that the activation function of the neurons is differentiable.

## 2.5. Genetic Algorithms

Genetic algorithms, which are algorithms for optimization, are stochastic search techniques that guide a population of solutions towards an optimum using the principles of evolution and natural genetics [29]. Their representation and operators characterize them. Basic genetic operators include reproduction, crossover and mutation [29].

A key concept for genetic algorithms is that of schemata, or building blocks . A schema is a subset of the fields of a chromosome set to particular values with the other fields left to vary. As originally observed in [30], the power of genetic algorithms lies in their ability to implicitly evaluate large numbers of schemata simultaneously and to combine smaller schemata into larger schemata [31].

Adopted from biological systems, genetic algorithms are based loosely on several features of biological evolution [23]. In order to work properly, they require five components (way of encoding solutions, evaluation function, way of initializing population, operators for reproduction and parameter settings). For details, see [31].

When the components of the GA are chosen appropriately, the reproduction process will continually generate better children from good parents, the algorithm can produce populations of better and better individuals, converging finally on results close to a global optimum. Additionally, GA can efficiently search large and complex (i.e., possessing many local optima) spaces to find nearly global optima [31]. In many cases, the standard operators, mutation and crossover, are sufficient for performing the optimization. Moreover, GAs are also capable of handling problems in which the objective function is discontinuous, non-differentiable, non-convex or noisy. Since the algorithms operate on a population instead of a single point in the search space, they climb many peaks in parallel and therefore reduce the probability of finding local minima [30].

## 3. Hypothesis

Scientists try to incorporate other methods into RBF network to better its outputs. For example, in [32] authors use genetic algorithms for creating "Evolving"

RBF – i.e. to automatically find the ideal number of hidden neurons. Montana in [31] uses genetic algorithms with multilayer perceptron neural network for weight adaptation.

First, RBF neural network will be combined with an unsupervised learning method for clustering called K-means. Since Kohonen [33] and Marcek [34] demonstrated that non-hierarchical clustering algorithms used with artificial neural networks could cause the better results of ANN, K-means will be used together with RBF in order to find out whether this combination can produce the effective improvement of this network in the domain of financial time series. The K-means will be used in the phase of non-random initialization of weight vector $w$ performed before the phase of network learning. In many cases it is not necessary to interpolate the output value by radial functions, it is quite sufficient to use one function for a set of data (cluster), whose center is considered to be a center of activation function of a neuron. The values of centroids will be used as the initialization values of weight vector $w$. Weights should be located near the global minimum of the error function Eq. (4), and the lower number of epochs is supposed be used for network training. We will use adaptive version of K-means.

Also, the back-propagation itself is a weakness of RBF. The convergence is slow and in addition it generally converges to any local minimum on the error surface, since stochastic gradient descent exists on the surface, which is not flat. Therefore, the back-propagation will be substituted by the genetic algorithm (GA) as an alternative learning technique in the process of weights adaptation. GAs are generally not bothered by local minima. The mutation and crossover operators can step from a valley across a hill to an even lower valley with no more difficulty than descending directly into the valley.

Finally, in this paper we also suggest a hybrid model. The reason to use this hybridization is to create a model with better forecasting properties. There exist quite a lot hybrid models in time series forecasting; for example in [35] authors combine Hidden Markov Models (HMM) with GARCH models to forecast time series, authors in [36] and [37] use a combination of Support Vector Machines (SVM) and genetic algorithms and [38] constructed a hybrid model composed of SVM and self-organizing maps (SOM) in time series prediction process. Our suggested hybrid model combines RBF neural network and moving average used for error modeling. Hence, in the next part of this paper, hypothesis that a combination of ANN and statistical model can produce a model with better properties, will be tested.

## 4. Pre-Experimental Procedures

### 4.1. Data and Model Validation

For our experiments daily close price of the USD/CAD currency was chosen. The interval was from 10/31/2008 to 10/31/2012, i.e. 1044 daily observations. The data was downloaded from the website `http://www.global-view.com/forex-trading-tools/forex-history`. Due to validation of the model, data were divided into two parts. The first part included 912 observations (from 10/31/2008 to 4/30/2012) and was used for training of the model. The second part of data (5/1/2012 to 10/31/2012) counting 132 observations, was used for model validation by making one-day-ahead ex-post forecast. These observations were not incorporated into model training (parameters of a model were not changing anymore) in order to find out the prediction power, as there is an assumption that if the model can handle to predict data from ex-post set, it will also be able to predict values of a currency pair in the future.

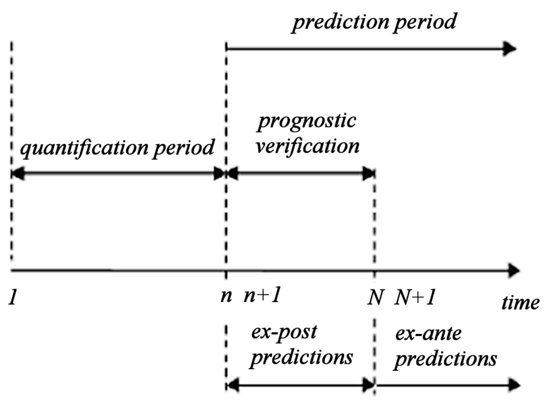

**Fig. 1:** Observations division of AUD/USD data.

### 4.2. Box-Jenkins Analysis

Box-Jenkins analysis was performed to make a comparison between statistical models and our tested neural network models. For statistical modelling, Eviews software was used. The ADF unit root test confirmed the hypothesis that the series was non-stationary. To confirm stationarity, the series was differentiated which is a necessary condition in Box-Jenkins modeling.

By analyzing autocorrelation and partial autocorrelation functions of first differences of USD/CAD, there was no significant coefficient in the series. Therefore, we proceeded to model this series as a pure GARCH process. The suitability for using stochastic volatility model was also accepted by performed heteroscedasticity test. ARCH test confirmed the series was heteroscedastic since the null hypothesis of homoscedasticity was rejected at 5 % and so the residuals were characterized by the presence of ARCH effect which is quite a frequent phenomenon at financial time series. Several models of ARCH [6] and GARCH [7], were estimated. The estimation of different models was only based on 912 in-sample observations, in order to make ex-ante predictions with remaining 132 observations. Marquardt optimization procedure was used for finding the optimal values of GARCH parameters; initial values of parameters were counted using Ordinary Least Squares (OLS) method and these values were then by iterative process consisted of 500 iterations. Convergence rate was set to 0.0001. After GARCH model had completed, the standardized residuals of this series were tested with Ljung-Box Q test in order to confirm there are no significant coefficients in residuals of this model. Our assumption was confirmed, hence according to statistical tests the model was correct. The final definition of the model is:

$$\log(h_t) = -0.172109 + 0.117148\frac{\varepsilon_{t-1}}{\sqrt{h_t}} + 0.037398\frac{\varepsilon_{t-1}}{\sqrt{h_t}} + 0.992135\log(h_{t-1}). \tag{8}$$

## 5. Experiments

The estimation of all models was only based on 912 observations, in order to make further comparisons with the predictions of the 132 remaining observations. In this paper, only one-step-ahead forecast were used, i.e. horizon of predictions was equal to one day. MSE (Mean Square Error) numerical characteristic was used for assessing models. The result of a given model is from the best neuron configuration (in every model we tested number of hidden neuron from 3 to 10 to find the best output results of the network). Experiment for every model configuration was performed 12 times; the best and worst results were eliminated and from the rest the mean and standard deviation were counted. Our created models of neural networks were then compared to standard statistical models.

### 5.1. RBF Neural Network

Own application of RBF neural network (implemented in JAVA with one hidden layer where we tested from three to ten processing neurons to achieve best results of network) was used. For every model, just the result with the best configuration is stated. The identity function was used as an activation function for the input layer and the output layer too. For the hidden layer the radial basis function was used as an activation function as it has been showed that it provides better accuracy than the perceptron network. In a normal case, the weights of neural network were initiated randomly – generated from the uniform distribution $< 0.1$). For the BP learning the learning rate was set to 0.001 to avoid the easy imprisonment in the local minimum. The number of epochs for each experiment with backpropagation was set to 5000 as this showed to be a good number for backpropagation convergence. The final results were taken from the best of 5000 epochs and not from the last epoch in order to avoid overfitting of the neural network. As raw (nonstandardized) data was used, we analyzed original nonstationary series for autocorrelation. As there was a strong dependence on the previous day (autocorrelation = 0.996), we used just one network input - the previous observation.

### 5.2. K-means Algorithm

K-means instead of random initialization of weights was used before they were adapted by BP. Coordinances of clusters were initiated as coordinances of randomly chosen input vector. After that, every input vector was assigned the nearest cluster. When this procedure has been done, the coordinates of clusters were recounted. This cycle was repeated 5000 times and the learning rate for the cluster adaptation was set to 0.001. The number of clusters was set to the number of hidden neurons.

In our experiments, the adaptive version of K-means will be used which is defined as follows:

- Random initialization of centroids in the dimension of the input vector.
- Introduction of the input vector $x_i$.
- Determination of the nearest from all centroids to a given input.
- Adaptation the coordinates of the centroid according to the rule: $c_{j'} = c^*_{j'} + \eta(x_i - c_{j'})$, where $j'$ is the nearest cluster to the introduced input, $\eta$ is a learning rate parameter.
- Termination of the algorithm if all inputs were processed or the coordinates of the cluster are not changing anymore.

### 5.3. Genetic Algorithm

Our own implementation of the genetic algorithm was used for weight adaptation. The chromosome length was set according to the formula: $D * s + s$, where $s$ is the number of hidden neurons and $D$ is the dimension of the input vector. A specific gene of the chromosome was a float value and represented a specific weight in the neural network. The whole chromosome represented weights of the whole neural network. The fitting function for evaluating the chromosomes was the mean square error function (MSE). The chromosome (individual) with the best MSE was automatically transferred into the next generation. The other individuals of the next generation were chosen as follows: By tournament selection (size of the tournament equaled to 100) 100 individuals were randomly chosen from the population. The fittest of them was then chosen as a parent. The second parent was chosen in the same way. The new individuals were then created by crossover operation. If the generated value from $\langle 0.1)$ was lower than 0.5 the weight of the first parent at the specific position was assigned to a new individual. Otherwise, a new individual received the weight of the second parent.

The mutation rate was set to 0.01. If performed, the specific gene (weight) of a chromosome was changed to a random value. The weight initialization for the first population of chromosomes was tested too. Testing intervals $\langle -1, 1\rangle$, $\langle -10, 10\rangle$ and $\langle -100, 100\rangle$ it was found out that the best results are presented when using weights generation from $-10$ to 10.

The size of the population and the number of generation for the genetic algorithm were set accordingly to the settings of back-propagation. In back-propagation there were 5000 cycles of the forward signal propagation plus 5000 cycles of backward error propagation. In GA the size of the population equalled to 1000 and 10 was the number of generations. As the operators of mutation and crossover do little computation, the computational demand is relatively the same.

### 5.4. Hybrid Model

Hybrid model is a combination of more independent models integrated into one model to produce only one output in specific time $t$. The main point in constructing hybrid models is to find out how to combine independent models in order to produce the best possible results.

In this work the following hybrid model was tested - a combination of RBF neural network and the error moving average model. The inspiration for the moving average part came from Box-Jenkins statistical models. We try to eliminate the error of the neural network by modeling the residuals of RBF just like moving averages of random part in Box-Jenkins ARIMA models. The hybridized model which was suggested is defined as follows:

$$y = RBF(x, w, v, s) + MA^*(q), \qquad (9)$$

$$y = \psi_2\Big(\sum_j v_j * \psi_1[\phi(\|x - w\|)]\Big) + \sum_{i=1}^{q} \varepsilon_{t-i}, \qquad (10)$$

$$y = \sum_{j=1}^{s} v_j * exp^{\frac{-\left\|x - w^j\right\|^2}{2\sigma_j^2}} + \sum_{k=1}^{q} e_{k-i}^{RBF}, \;\; j = 1..s, \qquad (11)$$

$$y_t = \sum_{j=1}^{s} v_j * exp^{\frac{-\sqrt{\sum_{i=1}^{k}(x_i^t - w_i^j)^2}^2}{2\sigma_j^2}} + \frac{1}{n}\sum_{n=1}^{q} e_{y_{t-n}}^{RBF}. \qquad (12)$$

## 6. Results and Discussion

The reason why the prediction qualities were applied on the validation set (ex-ante predictions) was the fact that an ANN can become so specialized for the training set that could lose accuracy in the test set. Therefore, it is obvious that even if the network provides acceptable results on the training set, it is not sure the results will not acceptable when they are applied to new data. Therefore, the estimation of all models was only based on 912 observations, in order to make further comparisons with the predictions of the 132 remaining observations. In this paper, only one-step-ahead forecast were used, i.e. horizon of predictions was equal to one day. We used MSE (Mean Square Error) numerical characteristic for assessing models. The result of a given model is from the best neuron configuration (in every model we tested number of hidden neuron from 3 to 10 to find the best output results of the network). Experiment for every model configuration was performed 12 times; the best and worst results were eliminated and from the rest the mean and standard deviation were counted.

First of all, from Tab. 1 it can be clearly seen (RBF network, one autoregressive input) that network with BP achieved the best results when there are 4 neurons in the hidden layer. On the other hand, the advanced methods for network learning (K-means + BP, GA) achieved the best results with 4 (GA), respectively 9 neurons (K-means + BP). However, when these advanced methods are used, the number of hidden neurons seem to not play an important role as the results where comparable . Following from that one can deduce that for remembering the relationships in this time series it is enough to use a smaller number of hidden neurons (three or four).

Secondly, the standard back-propagation algorithm for weights adaptation showed to be the great weakness of the neural network. The convergence was very slow and in addition, it generally converged to any local minimum on the error surface, since stochastic gradient descent existed on the surface which was not flat. It was due to the fact, that the gradient method does not guarantee to find optimal values of parameters and imprisonment in the local minimum is quite possible.

Bearing in mind the disadvantages of BP stated in the previous part of the paper, also other methods for network adaptation were tested– K-means, that was used in the phase of non-random initialization of weight vector w performed before the phase of network learning, and GA. It is no surprise that the RBF network combined with K-means or GA for weights adaptation provided significantly better results than the standard RBF (see Tab. 1). Moreover, besides lower MSE, another advantage of using a genetic algorithm or K-means upgrade is the consistency of predictions. The standard deviation of these methods is incomparably lower than the SD when the standard BP is used (see Tab. 1).

As for K-means, its biggest strength is in the speed of convergence of the network. Without K-means, it took considerably longer time to achieve the minimum. However, when the K-means was used to set the weights of the network before backpropagation, the time for reaching the minimum was much shorter. Therefore, the advantage of using K-means together with backpropagation is in the speed of adaptivity rather than in better predictions. Following from that one can say that in many cases it is not necessary to interpolate the output value by radial functions, it is quite sufficient to use one function for a set of data (cluster), whose center is considered to be a center of activation function of a neuron and the values of centroids can be used as an initialization values of weight vector w. The assumption used here was that weights should be located near the global minimum of the error function the (Eq. (4)). The advantage of this combination is that the lower number of epochs is supposed be used for network training. Moreover, Kmeans is quite simple to implement. However, one must bear in mind that Kmeans is a relatively efficient algorithm only in the domain of non-extreme values. Otherwise, other advanced non-hierarchical clustering algorithms must be used.

Having also tested GA in weights adaptation, it was found out the convergence was also considerably faster than at BP and therefore it was no surprise that sometimes the network converged only after 5 generations.

If we compare weights adaptation via GA and Kmeans plus backpropagation, the results are almost the same. Even though, K-means provided better results compared to GA, the differences are not very large. However, GA has a bigger potential to perform even better forecasts as there are more parameters needed to be optimized. Backpropagation, even though used with Kmeans, seemed to reach its global minimum as even with the higher number of epochs (we tested back-propagation up to 10000 cycles) the results were almost the same. The number of hidden neurons seemed to not play an important role as the results were comparable.

Also the number of inputs coming into the network in this paper was tested; the ANN with only one input provided better forecasts than the network with three autoregressive inputs (see Tab. 1).

For the RBF-SMA hybrid ANN model, the same strategy as for the standard ANN was used. With the given number of hidden neurons (step by step we tested the model with hidden neurons from three to ten) twelve testing procedures were performed. Just like for the standard model, we firstly trained the standard network, using either GA, BP or K-means plus BP, we then checked the predictive accuracy of the standard RBF using the last 133 observations which were not used for model training. Afterwards, the hybridization of the ex-ante predictions with moving averages of previous errors was performed. What for the value of parameter of the moving average, the values from one to one hundred were tested and we experimentally found out the best values for the tested data (for the majority of testing procedures the optimal value of moving average parameter was 44). Finally, just like for the standard RBF, from the best ten out of twelve experiments, the mean and standard deviation of the best results of RBF-SMA (having the optimal value of MA parameter) were counted. For every number of hidden neurons tested, the results are stated in the Tab. 1 which contains the results of out-of-sample predictions provided by the different models and optimization techniques, respectively.

To see the effectiveness or ineffectiveness of the suggested hybrid model, comparative analysis with individual model is usually performed. We provided the comparison with standard RBF network as well as the statistical ARIMA and GARCH model in order to show the prediction power of our suggested model. Table 2 states the final results of the numerical comparison of all tested and quantified models. Table 3 states the percentual comparison of our suggested model against the standard neural network.

Deducting from the Tab. 1 and Tab. 2, the network with only one input provided significantly better results in the validation set than the network with three inputs. The standard RBF provided the best outputs when it was combined with K-means and BP. The error of prediction at this network was a little bit lower compared to the statistical model; however these two models provided almost the same results.

Comparing the numerical (Tab. 2) as well as graphical results (Fig. 2 and Fig. 3), the suggested hybrid model improved the prediction power of the standard RBF considerably. Therefore, deducting from the performed experiments one can state that the application of our suggested new hybrid neural network model into the domain of exchange rates provides significantly bet-

**Tab. 1:** Prediction power of RBF-SMA model (back-propagation, one input).

| **Inputs** | **Neurons** | **Weights adaptation** | **RBF** | | **RBF-SMA** | |
|---|---|---|---|---|---|---|
| | | | **MSE** | **sd** | **MSE** | **sd** |
| Autoregressive(1) | 3 | BP | 0.0000282628 | 1.29939E-05 | 1.69513E-05 | 0.0000039062 |
| | | KM + BP | 0.0000175381 | 0.0000006224 | 0.0000137675 | 0.0000009931 |
| | | GA | 0.0000180929 | 0.0000016469 | 0.0000136146 | 0.0000003816 |
| | 4 | BP | 0.0000183763 | 0.0000028765 | 0.0000136485 | 0.0000005710 |
| | | KM + BP | 0.0000173006 | 0.0000004025 | 0.0000130549 | 0.0000003013 |
| | | GA | 0.0000176860 | 0.0000006219 | 0.0000137306 | 0.0000010974 |
| | 5 | BP | 0.0000299369 | 0.0000812952 | 0.0000168334 | 0.0000069884 |
| | | KM + BP | 0.0000174326 | 0.0000007575 | 0.0000133526 | 0.0000003885 |
| | | GA | 0.0000176925 | 0.0000016246 | 0.0000141386 | 0.0000011016 |
| | 6 | BP | 0.0000248756 | 0.0000105719 | 0.0000140990 | 0.0000016518 |
| | | KM + BP | 0.0000187115 | 0.0000024836 | 0.0000140002 | 0.0000011530 |
| | | GA | 0.0000205995 | 0.0000073265 | 0.0000139753 | 0.0000010496 |
| | 7 | BP | 0.000029955 | 0.0000381995 | 0.0000152401 | 0.0000018918 |
| | | KM + BP | 0.0000170959 | 0.0000002617 | 0.0000135883 | 0.0000004315 |
| | | GA | 0.0000265817 | 0.0000100553 | 0.0000160908 | 0.0000033735 |
| | 8 | BP | 0.0000530843 | 0.0000462909 | 0.0000161911 | 0.0000018501 |
| | | KM + BP | 0.0000169521 | 0.0000003200 | 0.0000133422 | 0.0000002243 |
| | | GA | 0.0000181709 | 0.0000016133 | 0.0000152679 | 0.0000030365 |
| | 9 | BP | 0.0000594814 | 0.0000611668 | 0.0000156977 | 0.0000018874 |
| | | KM + BP | 0.0000168649 | 0.0000002319 | 0.0000132936 | 0.0000003833 |
| | | GA | 0.0000290958 | 0.0000136948 | 0.0000174571 | 0.0000049429 |
| | 10 | BP | 0.0000842809 | 0.0000580551 | 0.0000163252 | 0.0000019133 |
| | | KM + BP | 0.0000179805 | 0.0000029834 | 0.0000139659 | 0.0000011918 |
| | | GA | 0.0000236821 | 0.0000093964 | 0.0000193432 | 0.0000056131 |
| Autoregressive(3) | 3 | BP | 0.0000246627 | 0.0000009284 | 0.0000161939 | 0.0000006213 |
| | | KM + BP | 0.0000294856 | 0.0000038188 | 0.0000206474 | 0.0000024092 |
| | | GA | 0.0000211109 | 0.0000028368 | 0.0000145220 | 0.0000008942 |
| | 4 | BP | 0.0000272034 | 0.0000076035 | 0.0000167076 | 0.0000014736 |
| | | KM + BP | 0.0000281149 | 0.0000049241 | 0.0000191500 | 0.0000028708 |
| | | GA | 0.0000211384 | 0.0000022806 | 0.0000150333 | 0.0000009834 |
| | 5 | BP | 0.0000409725 | 0.0000299478 | 0.0000184556 | 0.0000017546 |
| | | KM + BP | 0.0000256209 | 0.0000025080 | 0.0000173282 | 0.0000016944 |
| | | GA | 0.0000208047 | 0.0000018349 | 0.0000147753 | 0.0000008241 |
| | 6 | BP | 0.0000864375 | 0.0001034218 | 0.0000170064 | 0.0000011602 |
| | | KM + BP | 0.0000249070 | 0.0000011215 | 0.0000164002 | 0.0000004802 |
| | | GA | 0.0000221589 | 0.0000039497 | 0.0000153159 | 0.0000029622 |
| | 7 | BP | 0.0001348482 | 0.0000953912 | 0.0000182209 | 0.0000007447 |
| | | KM + BP | 0.0000271282 | 0.0000055004 | 0.0000165589 | 0.0000010142 |
| | | GA | 0.0000208281 | 0.0000024914 | 0.0000148913 | 0.0000014674 |
| | 8 | BP | 0.0001732475 | 0.0000906953 | 0.0000178474 | 0.0000005939 |
| | | KM + BP | 0.0000397440 | 0.0000458707 | 0.0000165625 | 0.0000007438 |
| | | GA | 0.0000219612 | 0.0000041218 | 0.0000154261 | 0.0000019254 |
| | 9 | BP | 0.0001491386 | 0.0000806291 | 0.0000178850 | 0.0000010715 |
| | | KM + BP | 0.0000323388 | 0.0000107404 | 0.0000182793 | 0.0000019037 |
| | | GA | 0.0000227303 | 0.0000034871 | 0.0000156919 | 0.0000012017 |
| | 10 | BP | 0.0000965767 | 0.0000678169 | 0.0000182020 | 0.0000006811 |
| | | KM + BP | 0.0000398751 | 0.0000456632 | 0.0000160730 | 0.0000011953 |
| | | GA | 0.0000212117 | 0.0000042826 | 0.0000150181 | 0.0000027353 |

**Tab. 2:** Final comparison of predictive qualities – best configurations (out-of-sample predictions).

| **Model** | **Regressor(s)** | **Weights Adaptation** | **Mean MSE *1** | **sd*2** |
|---|---|---|---|---|
| Standard ANN (RBF) | Autoregressive (1) | Back-propagation | $1.84 \cdot 10^{-5}$ | $2.88 \cdot 10^{-6}$ |
| | | K-means + Back-propagation | $1.69 \cdot 10^{-5}$ | $2.32 \cdot 10^{-7}$ |
| | | Genetic algorithm | $1.77 \cdot 10^{-5}$ | $6.22 \cdot 10^{-7}$ |
| | Autoregressive (3) | Back-propagation | $2.47 \cdot 10^{-5}$ | $9.28 \cdot 10^{-7}$ |
| | | K-means + Back-propagation | $2.49 \cdot 10^{-5}$ | $1.12 \cdot 10^{-6}$ |
| | | Genetic algorithm | $2.08 \cdot 10^{-5}$ | $1.83 \cdot 10^{-6}$ |
| Hybrid model | Autoregressive (1) | Back-Propagation | $1.36 \cdot 10^{-5}$ | $5.71 \cdot 10^{-7}$ |
| | | K-means + Back-propagation | $1.31 \cdot 10^{-5}$ | $3.01 \cdot 10^{-7}$ |
| | | Genetic Algorithm | $1.36 \cdot 10^{-5}$ | $3.82 \cdot 10^{-7}$ |
| | Autoregressive (3) | Back-propagation | $1.62 \cdot 10^{-5}$ | $6.21 \cdot 10^{-7}$ |
| | | K-means + Back-propagation | $1.64 \cdot 10^{-5}$ | $4.80 \cdot 10^{-7}$ |
| | | Genetic algorithm | $1.45 \cdot 10^{-5}$ | $8.94 \cdot 10^{-7}$ |
| AR(0)-EGARCH(1,1,1) | Conditional Variance (1) | Marquardt | $1.71 \cdot 10^{-5}$ | - |
| | | Berndt-Hall-Hall-Hausman | $1.71 \cdot 10^{-5}$ | - |

**Tab. 3:** Percentual improvement of the suggested hybrid model compared to the standard RBF.

| Inputs | Neurons | Improvement of suggested hybrid compared to the standard RBF [%] | | |
|---|---|---|---|---|
| | | BP | K-means + BP | GA |
| Autoregressive(1) | 3 | 40.022573 | 21.499478 | 24.751698 |
| | 4 | 25.727704 | 24.540767 | 22.364582 |
| | 5 | 43.770397 | 23.404426 | 20.087043 |
| | 6 | 43.32197 | 25.178633 | 32.157091 |
| | 7 | 49.123352 | 20.5172 | 39.466626 |
| | 8 | 69.499268 | 21.294707 | 15.976094 |
| | 9 | 73.609061 | 21.175933 | 40.001306 |
| | 10 | 80.630012 | 22.327521 | 18.321433 |
| Autoregressive(3) | 3 | 34.338495 | 29.974632 | 31.210891 |
| | 4 | 38.582677 | 31.886651 | 28.881562 |
| | 5 | 54.956129 | 32.366935 | 28.980951 |
| | 6 | 80.325206 | 34.154254 | 30.881497 |
| | 7 | 86.487843 | 38.960565 | 28.5038 |
| | 8 | 89.698322 | 58.327043 | 29.757481 |
| | 9 | 88.007799 | 43.475639 | 30.964835 |
| | 10 | 81.152804 | 59.691637 | 29.19898 |

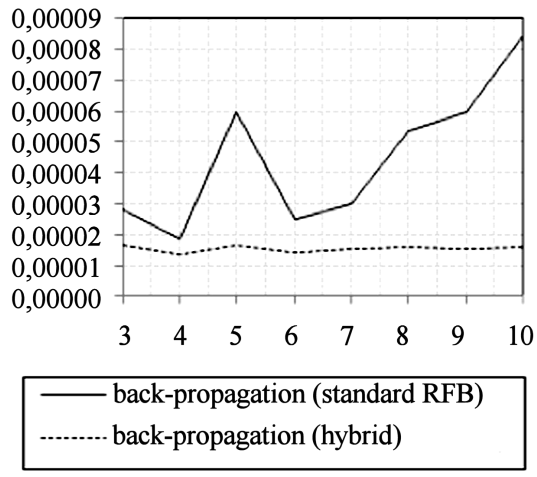

**Fig. 2:** Predictive accuracy of standard RBF model and RBF-SMA hybrid model (BP algorithm).

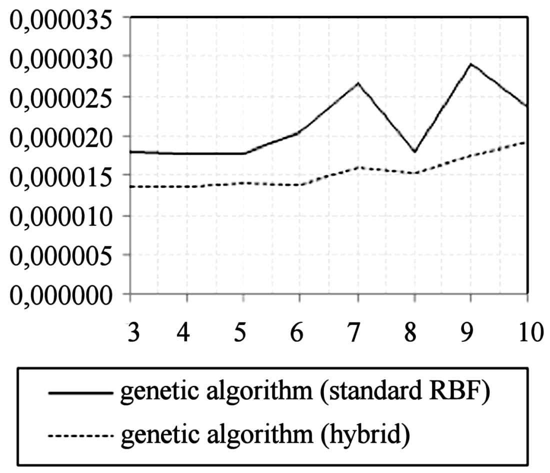

**Fig. 3:** Predictive accuracy of standard RBF model and RBF-SMA hybrid model (Kmeans + BP).

ter results than the standard RBF neural network as well as statistical ARIMA model.

## 7. Conclusion

In this paper we performed financial time series modeling with RBF neural networks as well as our suggested hybrid model. We used USD/CAD data that was divided into training set and validation due to model checking. Except for a standard ANN, we also combined an unsupervised learning method K-means and GA into the RBF in order to achieve better accuracy of the network. Both of the algorithms were used in the process of adapting weights of the network. The reason for incorporating other algorithms into the network was that the BP is considered a weakness of the RBF. Some of the drawbacks of BP include the scaling problem, the complexity problem, the slow convergence, the convergence to a local minimum etc. K-means was used in the phase of non-random initialization of weight vector $w$ before learning.

We also suggested the new hybrid neural network model in order to improve the prediction accuracy of the standard ANN. One can say we have used only the easiest and slow BP and not a more powerful version of BP. However, it is important to note that our main objective was to compare the standard neural network with our suggested hybrid model. Due to this we used only the standard BP in both models.

We performed experiments to find out that our suggested hybrid model based on RBF neural network had a significant predictive superiority over the statistical model as well as standard characteristics always overcame individual models (ANN, statistical model); the improvements were ranging from about 18 percent to more than 89 percent. Hence, the suggested hybrid showed to be a great improvement of the standard RBF neural network as we experimentally clearly proved that for the USD/CAD the hybrid model provided significantly better forecasts than the standard model of the RBF neural network and the statistical model and there was a clear benefit of better one-day-ahead forecasts. Following from these empirical findings for out-of-sample one-step-ahead forecasts, we believe that this model has a great potential in time series modeling.

## Acknowledgment

This paper was supported by European Social Fund within the project CZ.1.07/2.3.00/20.0296.

## References

[1] BROWN, R. G. *Smoothing Forecasting and Prediction of Discrete Time Series.* Englewood Cliffs, NJ: Prentice-Hall, 1963. ISBN 0-286-49592-2.

[2] HOLT, C. C. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting.* 2004, vol. 20, no. 1, pp. 5–10. ISSN 0169-2070. DOI: 10.1016/j.ijforecast.2003.09.015.

[3] WINTERS, P. R. Forecasting Sales by Exponentially Weighted Moving Averages. *Management Science.* 1960, vol. 6, no. 3, pp. 324–342. ISSN 1526-5501. DOI: 10.1287/mnsc.6.3.324.

[4] BOX, G., E. P. and G. M. JENKINS. *Time series analysis: forecasting and control.* San Fransisco, CA: Holden-Day, 2008. ISBN 978-0-470-27284-8.

[5] O'DONOVAN, T. M. *Short Term Forecasting: An Introduction to the Box-Jenkins Approach.* NY: New York, Wiley, 1983. ISBN 978-0-471-90013-9.

[6] ENGLE, R. F. Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica.* 1982, vol. 50, no. 4, pp. 987–1007. ISSN 1468-0262.

[7] BOLLERSHEV, T. Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics.* 1986, vol. 31, no. 3, pp. 307–327. ISSN 0304-4076. DOI: 10.1016/0304-4076(86)90063-1.

[8] HECHT-NIELSEN, R. *Neurocomputing.* Indianapolis: Addison-Wesley, 1990. ISBN 978-0201093551.

[9] HERTZ, J. A., A. S. KROGH and R. G. PALMER. *Introduction To The Theory Of Neural Computation.* Boulder: Westview Press, 1991. ISBN 978-0-201-51560-2.

[10] HIEMSTRA, C. and J. D. JONES. Testing for linear and nonlinear Granger causality in the stock price-volume relation. *The Journal of Finance.* 1994, vol. 49, no. 5, pp. 1639–1664. ISSN 1540-6261. DOI: 10.1111/j.1540-6261.1994.tb04776.x.

[11] DE GOOIJER, J. G. and R. J. HYNDMAN. 25 years of time series forecasting. *International Journal of Forecasting.* 2006, vol. 22, no. 3, pp. 443–473. ISSN 0169-2070. DOI: 10.1016/j.ijforecast.2006.01.001.

[12] HILL, T., L. MARQUEZ, M. O'CONNOR and W. REMUS. Artificial neural network models for forecasting and decision making. *International Journal of Forecasting.* 1994, vol. 10, no. 1, pp. 5–15. ISSN 0169-2070. DOI: 10.1016/0169-2070(94)90045-0.

[13] HWITE, H. Economic prediction using neural networks: the case of IBM daily stock returns. In: *IEEE International Conference on Neural Networks.* San Diego: IEEE, 1988, pp. 451–458. DOI: 10.1109/ICNN.1988.23959.

[14] MARCEK, D. Some Intelligent Approaches to Stock Price Modelling and Forecasting. *Journal of Information, Control and Management Systems.* 2004, vol. 2, no. 1, pp. 1–6. ISSN 1336-1716.

[15] THINYANE, H. and J. MILLIN. An Investigation into the Use of Intelligent Systems for Currency Trading. *Computational Economics.* 2011, vol. 37, no. 4, pp. 363–374. ISSN 0927-7099. DOI: 10.1007/s10614-011-9260-4.

[16] PARK, C. H. and S. H. IRWIN. The profitability of technical analysis: A review. In: *Social Science Research Network* [online]. 2004. Available at: `http://papers.ssrn.com/`.

[17] HASSAN, R., B. NATH and M. KIRLEY. A fusion model of HMM, ANN and GA for stock market forecasting. *Expert Systems with Applications.* 2007, vol. 33, no. 1, pp. 171–180. ISSN 0957-4174. DOI: 10.1016/j.eswa.2006.04.007.

[18] STERBA, J. and K. HILOVSKA. The Implementation of Hybrid ARIMA-Neural Network Prediction Model for Agregate Water Consumption Prediction. *Aplimat–Journal of Applied Mathematics.* 2010, vol. 3, no. 3, pp. 377–384. ISBN 1337-6365.

[19] ZHUANG, X.-F. and L.-W. CHAN. Volatility Forecasts in Financial Time Series with HMM-GARCH Models. In: *Intelligent Data Engineering and Automated Learning–IDEAL 2004.* Exeter: Springer, 2004, pp. 807–812. ISBN 978-3-540-22881-3. DOI: 10.1007/978-3-540-28651-6_120.

[20] CHIKHI, M., A. PEGUIN-FEISSOLLE and M. TERRAZA. SEMIFARMA-HYGARCH Modeling of Dow Jones Return Persistence. *Computational Economics*. 2013, vol. 41, no. 2, pp. 249–265. ISSN 1572-9974. DOI: 10.1007/s10614-012-9328-9.

[21] CHOUDHRY, R. and K. GARG. A Hybrid Machine Learning System for Stock Market Forecasting. *World Academy of Science, Engineering and Technology*. 2008, vol. 39, no. 3, pp. 315–318. ISSN 1543-5962.

[22] FALAT, L. and D. MARCEK. Volatility Forecasting in Financial Risk Management with Statistical Models and ARCH-RBF Neural Networks. *Journal of Risk Analysis and Crisis Response*. 2014, vol. 4, no. 2, pp. 77–95. ISSN 2210-8505. DOI: 10.2991/jrarc.2014.4.2.4.

[23] HORNIK, K. Some new results on neural network approximation. *Neural Networks*. 1993, vol. 6, no. 8, pp. 1069–1072. ISSN 0893-6080. DOI: 10.1016/S0893-6080(09)80018-X.

[24] HORNIK, K., M. STINCHCOMBER and H. WHITE. Multilayer feedforward networks are universal approximations. *Neural Networks*. 1989, vol. 2, no. 5, pp. 359–366. ISSN 0893-6080. DOI: 10.1016/0893-6080(89)90020-8.

[25] MACIEL, L. S. and R. BALLINI. Design a Neural Network for Time Series Financial Forecasting: Accuracy and Robustness Analysis. In: *Computer Science & Engineering*. 2008. Available at: `http://www.unr.edu/cse/`.

[26] DARBELLAY, G. A. and M. SLAMA. Forecasting the short-term demand for electricity: Do neural networks stand a better chance? *International Journal of Forecasting*. 2000, vol. 16, no. 1, pp. 71–83. ISSN 0169-2070. DOI: 10.1016/S0169-2070(99)00045-X.

[27] CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems (MCSS)*. 1989, vol. 2, no. 4, pp. 303–314. ISSN 1435-568X. DOI: 10.1007/BF02551274.

[28] BRYSON, A., E. and Y.-Ch. HO. *Applied optimal control: optimization, estimation, and control*. London: Taylor & Francis, 1969. ISBN 978-0891162285.

[29] VISHWAKARMA, D. D. Genetic Algorithm based Weights Optimization of Artificial Neural Network. *Network International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*. 2012, vol. 1, no. 3, pp. 206–211. ISSN 2278-8875.

[30] HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. Michigan: University of Michigan Press. 1975. ISBN 9780262581110.

[31] MONTANA, D. J. and L. DAVIS. Training feedforward neural networks using genetics algorithms. In: *Proceedings of the 11th International Joint Conference on Artificial Intelligence*. San Francisco: Morgan Kaufmann Publishers Inc., 1989, pp. 762–767.

[32] RIVAS, V. M., J. J. MERELO, P. A. CASTILLO, M. G. ARENAS and J. G. CASTELLANO. Evolving RBF neural networks for time-series forecasting with EvRBF. *Information Sciences*. 2004, vol. 165, no. 3–4, pp. 207–220. ISSN 0020-0255. DOI: 10.1016/j.ins.2003.09.025.

[33] KOHONEN, T. *Self-organising maps*. Berlin: Springer, 1995. ISBN 3-540-58600-8.

[34] MARCEK, D. and M. MARCEK. *Neuronove siete a ich aplikacie*. Zilina: EDIS–Vydavatelstvo Z, 2006. ISBN 80-8070-497-X.

[35] ZHUANG, X.-F. and L.-W. CHAN. Volatility Forecasts in Financial Time Series with HMM-GARCH Models. In: *Intelligent Data Engineering and Automated Learning–IDEAL 2004*. Exeter: Springer, 2004, pp. 807–812. ISBN 978-3-540-22881-3. DOI: 10.1007/978-3-540-28651-6_120.

[36] CHOUDHRY, R. and K. GARG. A Hybrid Machine Learning System for Stock Market Forecasting. *World Academy of Science, Engineering and Technology*. 2008, vol. 2, no. 3, pp. 242–245. ISSN 1543-5962.

[37] HONG, W., P. PAI, S. YANG and R. THENG. Highway traffic forecasting by support vector regression model with tabu search algorithms. In: *Proceeding of Internationall Joint Conference on Neural Networks*. Vancouver: IEEE, 2006, pp. 1617–1621. ISBN 0-7803-9490-9. DOI: 10.1109/IJCNN.2006.246627.

[38] CAO, L. and E. H. F TAY. Modified support vector machines in financial time series forecasting. *Neurocomputing*. 2002, vol. 48, no. 1–4, pp. 847–861. ISSN 0925-2312. DOI: 10.1016/S0925-2312(01)00676-2.

## About Authors

**Lukas FALAT** was born in 1986. He received his M.Sc. from University of Zilina in 2011. His research interests include feedforward neural networks, RBF networks, hybrid neural networks, time series modeling.

**Dusan MARCEK** is a Professor at VSB–Technical University of Ostrava. His research interests include RBF neural networks, fuzzy and soft neural networks. He has been invited to many world conferences in countries such as China, Singapore, Hong Kong, Turkey etc.