Data Stream Mining & Processing

PROCEEDINGS of the

2018 IEEE Second International Conference on

Data Stream Mining & Processing (DSMP)

IEEE Ukraine Section (Kharkiv)
SP/AP/C/EMC/COM
Societies Joint Chapter

IEEE Ukraine Section (West)
AP/ED/MTT/CPMT/SSC
Societies Joint Chapter

August 21–25, 2018

Lviv, Ukraine

There is an unknown target addiction – reflection y*:X →Y. The value is known only on known states of the robot on the training set $X_m=\{(x_1,y_1),\dots,(x_m,y_m)\}$.

It is necessary to develop an algorithm a:X→Y for classifying the robot's state Y according to sensor's reading x∈X in real time-domain. In our case, the set of classes is Y={0,1,2,3,4,5,6,7}.

A number of experiments were carried out with a three-axis MEMS gyroscope rigidly fixed to the robot body. Gyroscope allows you to track the robot's precise execution of the prescribed actions, possible features of its movement. Every action is a certain state.

III. Results & Analysis

The results of measurements are shown in Fig.2-4 and visualize the sensors measure in the process of robot moving. To find the clustering algorithm which is support real-time work we consider three axis of the gyroscope.

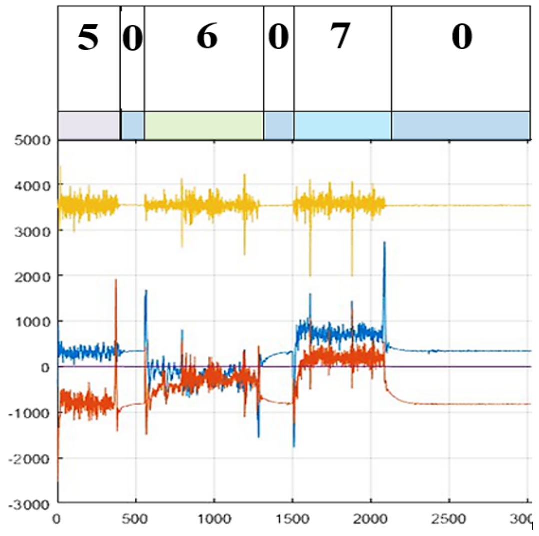

Fig. 2. " Procedure "moving forward - stop - rotation counterclockwise - stop - clockwise rotation - stop"

Figure 2 illustrates the gyroscope readings captured by the three axes. A detail visual analysis of figure 2 reveals that the relative magnitudes of the sub readings of the gyroscope could be used for event classification. For example from point 1 to point 400, robot was moving forward. And from 401 to 520 robot standing – this is indicated by the very low gyroscope values.

However it is also clear that coming up with manually defined thresholds for three sensor readings that will allow the classification of the seven events will be still a complex task. Further the raw signals captured by the sensors are noisy and will therefore have to be cleaned prior to further analysis.

The robot motion activity recognition system has to decide which of the seven events have effectively caused the measured values of the features based on real signals, which are fed from sensors. This is a general classification problem that can be dealt with by a large range of algorithms, such as logics, k-nearest Neighbor approaches, Support Vector Machines (SVMs), Artificial Neural networks [3,4], Decision Trees or Bayesian Techniques [5]. Our work focused on finding an algorithm which are realize a classification of robot motion in real-time domain.

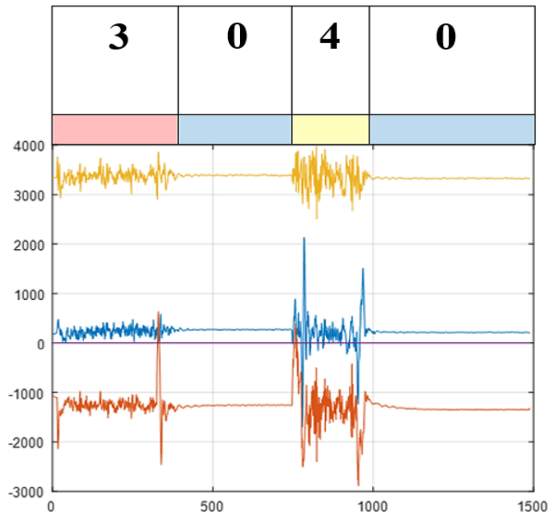

Fig. 3. Procedure "backwards motion on the slope – stop – forward motion from the slope, stop»

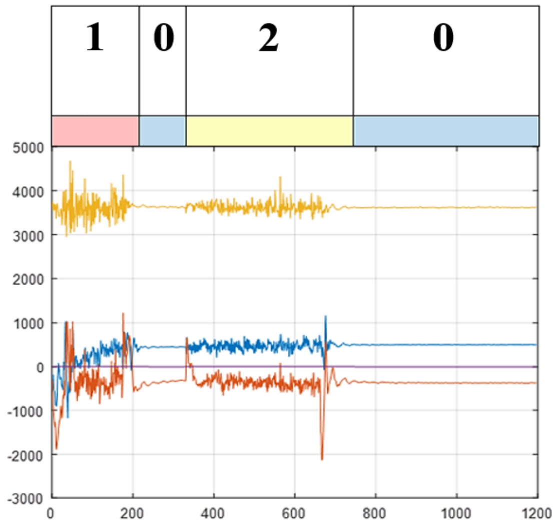

Fig. 4. Procedure "forward motion on the slope – stop – backwards motion from the slope - stop"

The thee time-domain features (three axis of the gyroscope) were used to train machine learning algorithm. The mean, standard deviation, minimum, and maximum of signal in the running window also were used as features. But using more features did not improve the quality of the classification

We examined the performance of some supervised learning algorithms and singled out most appropriate among them: Support Vector Machines (Linear SVM) [3], k-nearest neighbors algorithm (Medium KNN, Weighted KNN) [4], Boosting algorithm [6], Classification Trees (Simple Tree, Medium Tree) and Ensemble (Bagged trees) [7, 8].

Gyroscope signals from robot are sufficient to classification. Weighted KNN and Bagged trees performed slightly better than other three algorithms (the classification accuracy about 89%).

The evaluation of the quality of the trained models was carried out by such criteria as accuracy, confusion matrix,

IEEE Second International Conference on Data Stream Mining & Processing
August 21-25, 2018, Lviv, Ukraine

A Hybrid Neuro-Fuzzy Model for Stock Market Time-Series Prediction

Alexander Vlasenko
Department of Artificial Intelligence
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
alexander.vlasenko86@gmail.com

Nataliia Vlasenko
Department of Informatics and Computer Engineering
Simon Kuznets Kharkiv National University of Economics
Kharkiv, Ukraine
gorohovatskaja@gmail.com

Olena Vynokurova
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
IT Step University
Lviv, Ukraine
vynokurova@gmail.com

Marta Peleshko
Department of Monitoring and Fire Prevention
Lviv State University of Life Safety
Lviv, Ukraine
marta.peleshko@gmail.com

***Abstract*— In this paper we propose a hybrid five-layer neuro-fuzzy model and a corresponding learning algorithm with application in stock market time-series prediction tasks. The key difference between classical ANFIS architecture and the proposed model is in the fourth layer – multidimensional Gaussian functions are used instead of polynomials in order to achieve better computational performance and representational abilities in processing highly nonlinear volatile data. The experimental results have shown the clear advantages of the described model and its learning.**

Keywords*— *time series, neuro-fuzzy, membership function, Gaussian, prediction

I. Introduction

Time series prediction is the one of the most common complex practical problems which appears in various applied domains. Among different types of time series financial data (e.g. currency exchange, stock and derivatives market) could be distinguished by their highly nonlinear, nonstationary, volatile and chaotic nature and high-frequency dynamics.

For decades' statistical models have dominated the field of quantitative time-series analysis. The most popular among them are Autoregressive Moving Average (ARMA), Autoregressive Integrated Moving Average (ARIMA) etc. However, such models require time-series to be stationary and this is achieved through different differencing techniques, effectiveness of which is disputed due to highly nonstationary nature of real life time-series [1].

Except classical statistical models different computational intelligence techniques have been applied to the time series forecasting problems. Among them neuro-fuzzy models have gained popularity due to their universal approximation abilities and robustness combined with good computational performance and inherited from ANNs learning capabilities. They have been successfully used in the many forecasting tasks in different domains e.g. electric networks loading [2, 3], finances [4, 5, 6, 7] and others.

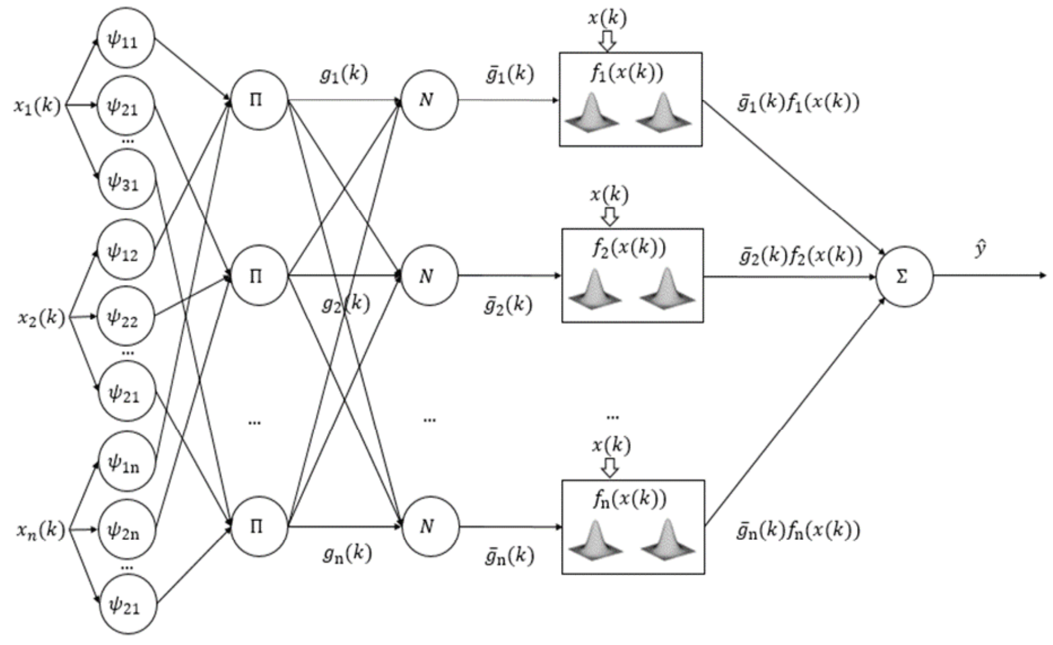

Fig. 1. The general architecture of the proposed model.

II. Architecture and inference

The proposed model is based on the classical ANFIS [8] model and comprises five layers. Fig.1 depicts a general architecture of the introduced model

The first layer is responsible for the fuzzification of the input variables represented by a n -dimensional vector $x(k) = (x_1(k), x_2(k),..., x_n(k))^T$. For the prediction task the input vector means the historical gap of the observed variable. Each input parameter is processed by the h^{ψ} membership functions, hence total amount of the first layer membership functions is $h^{\psi} \times n$. In current study we've used the Gaussian membership function, but in general case other common membership function types (e.g. triangular or generalized bell shaped) are acceptable. The Gaussian membership function has the following form:

$$\psi_{jl}\left(x_i(k)\right) = \exp\left(-\frac{\left(x_i(k) - c_{jl}^{\phi}\right)}{2\sigma_{ji}^2(k)}\right) \qquad (1)$$

Where $x(k)$ is an input vector, c_{jl}^{φ} – a centre of current Gaussian and σ_{jl} is a width parameter. The argument of the exponential function $-\frac{\left(x_i(k) - c_{jl}^{\phi}\right)}{2\sigma_{ji}^2(k)}$ is a quadratic function of x_i . The main advantages of this membership function is relatively small amount of parameters and proneness to outliers.

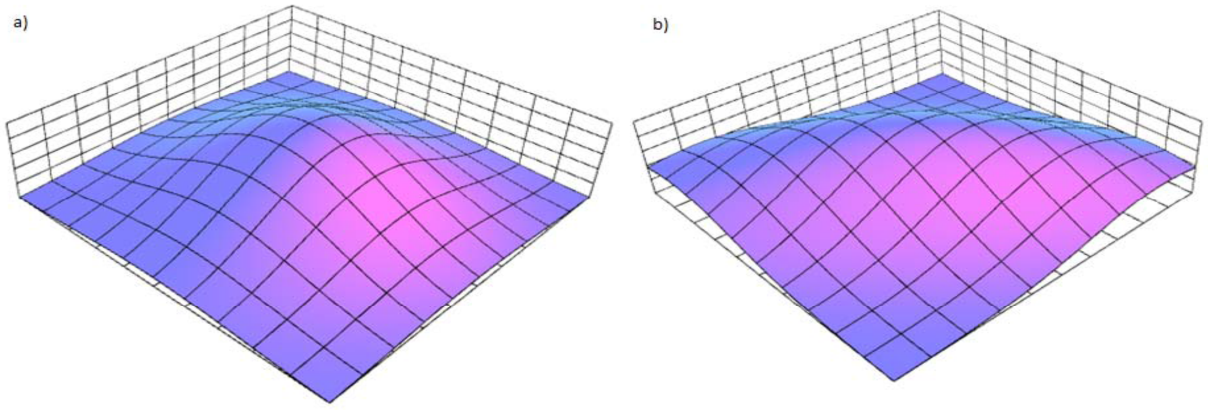

Fig. 2. The examples of multidimensional Gaussain with different covariance matrices: a) – with diagonal matrix; b – with matrix $\begin{bmatrix} 0.9 & 0.6 \\ 0.7 & 0.9 \end{bmatrix}$.

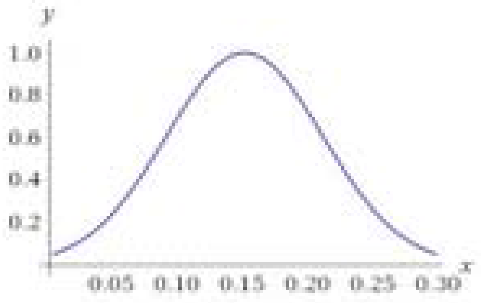

Fig. 3. The example of the first layer membership function with centre 0.15 and with width 0.06.

The second layer represents an aggregation of the antecedent premises values. It consists of h^{ψ} multiplier units which implement algebraic product fuzzy T-norm:

$$g_j(k) = \prod_{i=1}^{n} \psi_{jl}\left(x_i(k)\right) \qquad (2)$$

The third layer is non-parametrised and responsible for normalization. It also has h^{ψ} units which output is computed by the following formula:

$$\bar{g}_j(k) = \frac{g_j(k)}{\sum_{j=1}^{h^{\psi}} g_j(k)} = \frac{\prod_{i=1}^{n} \psi_{jl}\left(x_i(k)\right)}{\sum_{i=1}^{h^{\psi}} \prod_{i=1}^{n} \psi_{jl}\left(x_i(k)\right)} \qquad (3)$$

This is necessary for satisfying the Ruspini partitioning condition:

$$\sum_{j=1}^{h^{\psi}} \bar{g}_j(k) = 1 \qquad (4)$$

The forth layer is presented by multidimensional Gaussian consequent functions $\varphi_{je}(x(k))$ and their weights p . Multidimensional Gaussians are used instead of the standard TSK/ANFIS polynomials:

$$\varphi_{je}(x(k)) = \exp\left(-\frac{\left(x(k) - c_{je}^{\varphi}\right)^T Q_{je}^{-1}\left(x(k) - c_{je}^{\varphi}(k)\right)}{2}\right) \qquad (5)$$

where $x(k)$ is an input vector, c_{je}^{φ} – a vector which represents the centre of the current Gaussian and Q_{je} – covariance (receptive field) matrix, $Q_{je} \in S_{++}^{n}$. In this case a quadratic function from (1) becomes a quadratic form of the whole input vector $x(k)$.

Multidimensional Gaussian is a powerful tool to represent data which are not distributed evenly on the main axes. Hence the forth layer output is:

$$f_j(x(k)) = \sum_{e=1}^{h^{\varphi}} p_{je}\varphi_{je}(x(k)) \tag{6}$$

where h^{φ} is a number of multidimensional Gaussian functions for each unit f_j.

Fifth layer is non-parametrized and produces overall model output as a sum of its inputs:

$$\hat{y} = \sum_{j=1}^{h^{\varphi}} g_j f_j(x(k)) \tag{7}$$

In matrix form it could be rewritten as:

$$\hat{y} = p^T f(x(k)) \tag{8}$$

where $x(k)$ is an input vector, p -weights vector and $f(x(k))$ is a vector of normalised consequent functions values:

$$f(x(k)) = (\bar{g}_1\varphi_{11}(x(k))\ldots\bar{g}_{h^{\psi}}\varphi_{h^{\psi}h^{\varphi}}(x(k))) \tag{9}$$

where h^{ψ} is a number of functions in the first layer and h^{φ} - number of multidimensional Gaussians in the fourth layer for each normalized output $\bar{g}_1$.

III. Learning algorithms

Learning process in proposed model consists of adjusting weights vector p and tuning Multidimensional Gaussian functions parameters - both centres c_{jl}^{φ} and matrices Q_{jl}. First layer membership functions centres c_{jl}^{φ} are distributed equidistantly on initialization and they are not tuned during learning. Weights p are initialised randomly in range [-0.1;0.1] and their learning is achieved through the Kaczmarz iterative method:

$$p(k+1) = p(k) + \frac{y(k) - p^T f(x(k))}{f^T(x(k))f(x(k))} f(x(k)) \tag{10}$$

where $p(k)$ is a weights matrix represented as a vector, $y(k)$ -reference signal, $p^T f(x(k))$ - overall model output according to (7).

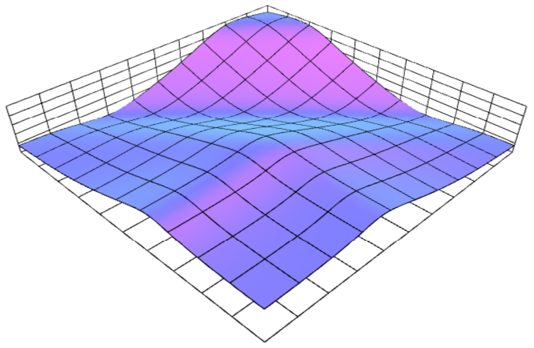

Fig. 4. The example of error surface in weights space.

Forth layer Gaussians learning performed by the first-order gradient backpropagation procedure based on the standard mean square error criterion:

$$E = \frac{1}{N}\sum_{k=1}^{N}(y(k) - \hat{y}(k))^2 \tag{11}$$

where $y(k)$ - reference signal value, $\hat{y}(k)$ - prognosis signal value and N is the training set length.

The Q_{jl}^{-1} matrices are initialized as identity matrices and their learning can be written in the following way:

$$\begin{cases} Q_{jl}^{-1}(k+1) = Q_{jl}^{-1}(k) + \lambda_Q \dfrac{\tau_{jl}^{Q}(k)e(k)}{\eta_q(k)} \\ \eta_Q(k+1) = \beta_Q\eta_Q(k) + Tr\left(\tau_{jl}^{Q^T}\tau_{jl}^{Q}\right) \end{cases} \tag{12}$$

where λ_Q is a learning step and β_Q is a momentum hyperparameters, τ_{jl}^{Q} is a vector of values back propagated for each multidimensional Gaussian.

The centres c_{jl}^{φ} are placed equidistantly on initialization step and then tuned by the formula below.

$$\begin{cases} c_{jl}^{\varphi}(k+1) = c_{jl}^{\varphi}(k) + \lambda_c \dfrac{\tau_{jl}^{c}(k)e(k)}{\eta_c(k)} \\ \eta_c(k+1) = \beta_c\eta_c(k) + \tau_{jl}^{c^T}\tau_{jl}^{c} \end{cases} \tag{13}$$

where λ_c is a learning step and β_c is a momentum hyperparameters, τ_{jl}^{c} is a vector of back propagated values.

IV. Experimental results

Proper and unbiased comparison of the different forecasting models is a complex task [9]. The proposed model and learning algorithm have shown good performance in real-life stock market datasets - daily log returns of IBM and Cisco. We have compared performance and prediction accuracy to ANN based on bipolar sigmoid activation functions and resilient backpropagation learning algorithm. RMSE and SMAPE criteria were used.

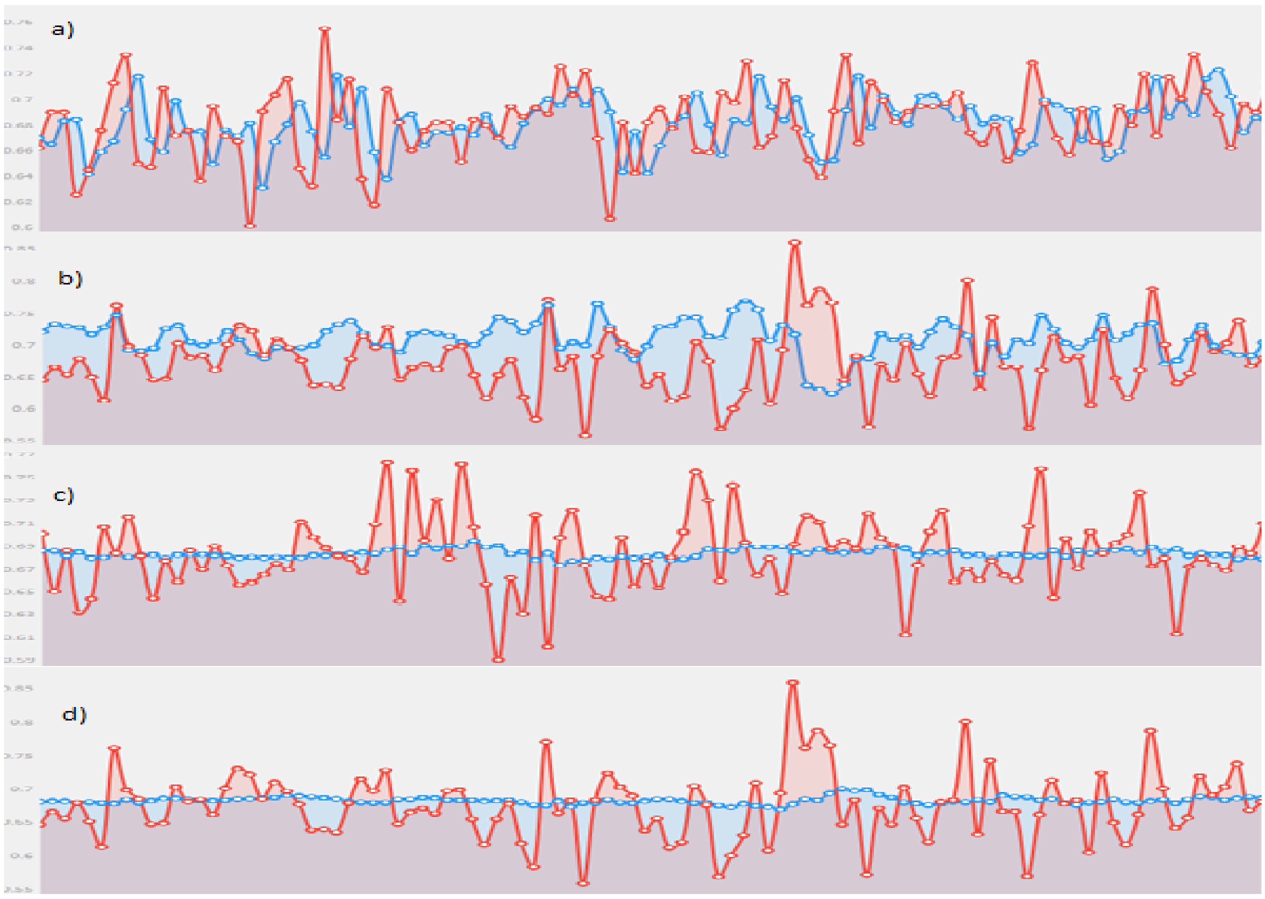

Fig. 5. Experiments on Cisco daily returns plots: a) proposed model – learning; b) proposed model – verification; c) bipolar sigmoid network – learning; d) bipolar sigmoid network – verification

The best results for the introduced model were achieved with $h^{\varphi}=2$, $\lambda_c=0.87$, $\beta_c=1.01$, $\lambda_Q=0.81$, $\beta_Q=1.02$. The bipolar sigmoid network was trained with alpha value 0.4, learning rate 0.67 and 100 epochs.

Model	**Cisco daily log returns dataset results**		
	Execution time, ms	***RMSE, %***	***SMAPE, %***
Proposed model	442	3.237	3.53
Bipolar Sigmoid Network	2668	3.31	3.6

V. Conclusion

In the paper a novel MIMO neuro-fuzzy model with multidimensional Gaussian functions in consequent layer is introduced. This solution allows to handle complex non-linear data, which is demonstrated on the stock market time-series prediction tasks.

References

[1] J. J. F. Commandeur, and S. J. Koopman, An Introduction to State Space Time Series Analysis. Oxford University Press. 2007.

[2] Ye. Bodyanskiy, S. Popov, and T. Rybalchenko, "Multilayer neuro-fuzzy network for short term electric load forecasting," Computer Science–Theory and Applicationsm pp, 339-348, 2008.

[3] P. Otto, Ye. Bodyanskiy, and V. Kolodyazhniy, "A new learning algorithm for a forecasting neuro-fuzzy network." Integrated Computer-Aided Engineering, vol, 10(4), pp, 399-409, 2003.

[4] Ye. Bodyanskiy, and S. Popov, "Neural network approach to forecasting of quasiperiodic financial time series,." European Journal of Operational Research, vol, 175(3), pp, 1357-1366, 2006.

[5] E. Hadavandi, H. Shavandi, and A. Ghanbari, “Integration of genetic fuzzy systems and artificial neural networks for stock price forecasting,” Knowledge-Based Systems, vol. 23, no. 8, pp. 800-808, 2010.

[6] A. Esfahanipour, and W. Aghamiri, “Adapted neuro-fuzzy inference system on indirect approach TSK fuzzy rule base for stock market analysis,” Expert Systems with Applications, vol. 37, no. 7, pp. 4742-4748, 2010.

[7] G. S. Atsalakis, and K. P. Valavanis, “Forecasting stock market short-term trends using a neuro-fuzzy based methodology,” Expert Systems with Applications, vol. 36, no. 7, pp. 10696-10707, 2009.

[8] J. S. R. Jang, “ANFIS: adaptive-network-based fuzzy inference system,” IEEE transactions on systems, man, 2and cybernetics, vol. 23, no. 3, pp. 665-685, 1993.

[9] S. F. Crone, M. Hibon, and Konstantinos Nikolopoulos. “Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction,” International Journal of Forecasting, vol. 27, pp. 635-660, 2011.