




Article

# A Novel Neuro-Fuzzy Model for Multivariate Time-Series Prediction <sup>†</sup>

Alexander Vlasenko <sup>1,\*</sup>, Nataliia Vlasenko <sup>2</sup>, Olena Vynokurova <sup>3,4</sup> and Dmytro Peleshko <sup>3</sup>

<sup>1</sup> Department of Artificial Intelligence, Faculty of Computer Science, Kharkiv National University of Radio Electronics, 61166 Kharkiv, Ukraine

<sup>2</sup> Department of Informatics and Computer Engineering, Faculty of Economic Informatics, Simon Kuznets Kharkiv National University of Economics, 61166 Kharkiv, Ukraine; gorohovatskaja@gmail.com

<sup>3</sup> Information Technology Department, IT Step University, 79019 Lviv Oblast, Ukraine; dpeleshko@gmail.com

<sup>4</sup> Control Systems Research Laboratory, Kharkiv National University of Radio Electronics, 61166 Kharkiv, Ukraine; vynokurova@gmail.com

\* Correspondence: alexander.vlasenko86@gmail.com; Tel.: +38-099-956-8084

<sup>†</sup> This paper is an extended version of the paper: A Hybrid Neuro-Fuzzy Model for Stock Market Time-Series Prediction. In Proceedings of the 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 21–25 August 2018; Lviv Polytechnic Publishing House: Lviv, Ukraine, 2018; pp. 352–356.

Received: 6 November 2018; Accepted: 6 December 2018; Published: 8 December 2018



**Abstract:** Time series forecasting can be a complicated problem when the underlying process shows high degree of complex nonlinear behavior. In some domains, such as financial data, processing related time-series jointly can have significant benefits. This paper proposes a novel multivariate hybrid neuro-fuzzy model for forecasting tasks, which is based on and generalizes the neuro-fuzzy model with consequent layer multi-variable Gaussian units and its learning algorithm. The model is distinguished by a separate consequent block for each output, which is tuned with respect to the its output error only, but benefits from extracting additional information by processing the whole input vector including lag values of other variables. Numerical experiments show better accuracy and computational performance results than competing models and separate neuro-fuzzy models for each output, and thus an ability to implicitly handle complex cross correlation dependencies between variables.

**Keywords:** time series; neuro-fuzzy; membership function; backpropagation; Kachmarz method; Gaussian; prediction

## 1. Introduction

Time-series represent data points ordered in the time domain and they arise in many fields including economics, engineering, sociology, and medicine. Mathematical tools for time-series forecasting can crucially improve the decision-making process. Statistical models have dominated the field of quantitative time-series analysis for decades. These models are predominantly parametric and therefore require time-series to be weakly stationary, which is achieved through various differencing techniques, whose effectiveness is disputed [1,2] due to the highly nonstationary and nonlinear nature of real-life time-series.

The computational intelligence (CI) approach proposes nature-inspired models and methods that have been competing against classical statistical approaches in many applied domains. Among CI principal techniques, Artificial Neural Networks (ANN) are the most known and widely used. Many neural models have been applied to financial time-series forecasting [3–5]. Crone et al. [4]

proved the ability of such networks to represent complex data patterns, including highly frequent, seasonal, and short time-series.

Fuzzy logic and fuzzy sets theory have been extended to a variety of mathematical objects by introducing a gradual degree of membership and have influenced uncertainty analysis [6,7]. Rule-based fuzzy models have been applied to various domains including stock market forecasting [8,9]. However, they have some essential drawbacks primarily caused by their dependence on expert knowledge or complex rule extraction methods.

Neuro-fuzzy systems combine the strengths of ANNs and fuzzy inference systems and have been successfully applied to a variety of problems. Their advantages are especially visible in rapidly changing domains with a high degree of uncertainty, e.g., routing guidance and planning [10,11], electric networks loading forecasting [12,13], complex logistics [14], crude oil blending [15], human resources portfolio management [16], and many others. Kar et al. [17] provided a brief review of neuro-fuzzy model applications including forecasting. The Adaptive Network Inference System (ANFIS), proposed by Jang [18] was the first successfully applied neuro-fuzzy model and the majority of the later models are based on it. Recently, neo-fuzzy systems [19–21], which are based on zero-order additive models, have been gaining popularity in various domains.

In the field of stock market prediction, examples of successfully applied neuro-fuzzy models include the ANFIS model with a modified Levenberg-Marquardt learning algorithm for Dhaka Stock Exchange day closing price prediction [22] and an ANFIS model based on an indirect approach and tested on Tehran Stock Exchange Indexes [23]. Rajab & Sharma [24] proposed an interpretable neuro-fuzzy approach to stock price forecasting applied to various exchange series. Neuro-fuzzy prediction models have been combined with genetic algorithms [25], wavelet transform [26], support vector machines (SVM) [27], and recurrent connections in order to use dynamic memory [28,29]. Such combined models could show high effectiveness, but they suffer from the additional costs of extra clustering procedures, layers, and hyper-parameters.

Generally, neuro-fuzzy systems require many rules to cover complex nonlinear relations. To solve this issue for scenarios where input variables show a certain degree of interdependence and correlation, in Ebadzadeh & Salimi-Badr [30], multivariable Gaussian functions were used in a neuro-fuzzy model for efficient handling of correlated input space regions, but prediction performance was measured on a chaotic time-series, where the behavior may drastically vary from real-life data. Not much attention was paid to the data noise handling.

The aforementioned works primarily focused on a single output scenario, so developing a compact model with a memory-wise architecture and effective learning that can handle complex cross-correlation dependencies between different input times series is a necessary task. The learning procedure should combine generalization and smoothing abilities in order to work on small and noisy datasets.

In this work, we present a novel neuro-fuzzy model and a corresponding learning procedure for time-series forecasting. This model has three advantages: (1) achieves better accuracy through using the representational abilities of the multivariable Gaussian functions in the fourth layer; (2) shows good computational performance, which is achieved by using a stochastic gradient optimization procedure for tuning consequent layer units and an iterative projective algorithm for tuning their weights; and (3) has a reasonable amount of hyperparameters that can be chosen.

The remainder of this paper is organized as follows: Section 2 describes the data set that was used for numerical experiments, Section 3 is devoted to the proposed model architecture and learning, and Section 4 contains experimental results.

## 2. Data Description

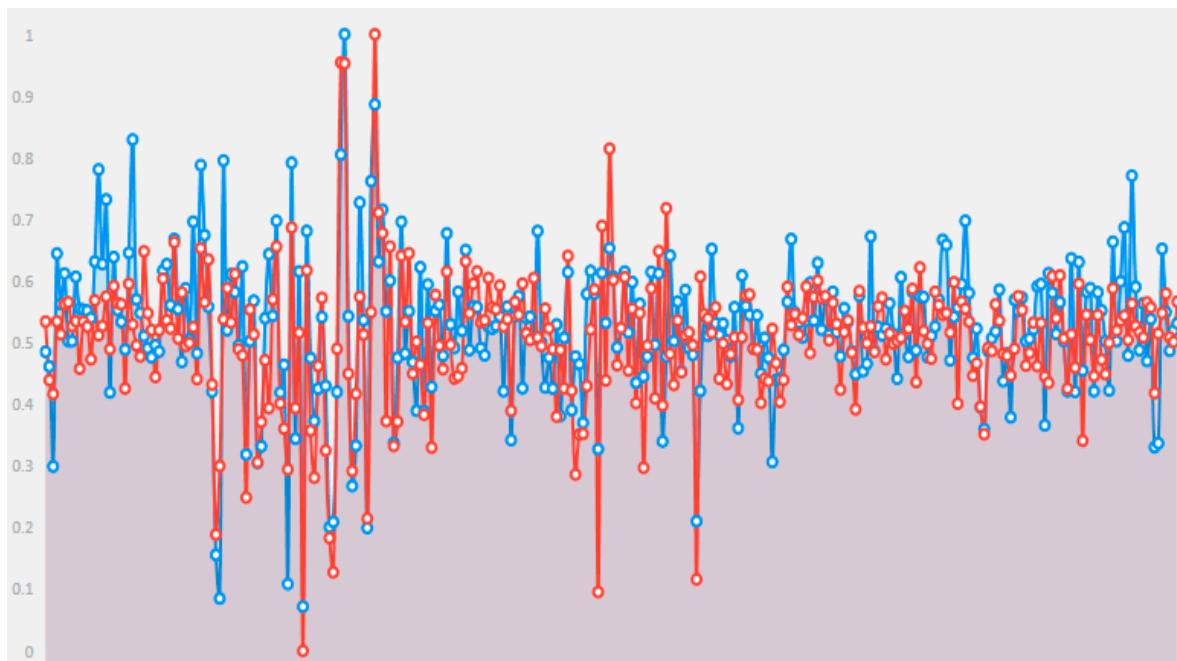
In order to verify our model performance, we used the log returns of IBM stock and the S&P 500 index from January 1926 to December 1999 with 888 monthly records as the dataset. Correspondingly, the dataset had two columns with numerical values: IBM represents the log returns of IBM stock, and

SP column contains the S&P 500 index values. These two returns comprise a bivariate concurrently correlated time series [31]. We chose this dataset as a framework due to its well-studied statistical properties and real-life nature. The original dataset can be found in Tsay [31,32] and Table 1 contains a randomly chosen block of 10 lines.

**Table 1.** Records example of the IBM stock and the S&P 500 index dataset.

IBM	SP
−8.24798	−4.06866
6.67236	−4.81089
4.73701	7.43094
0.70948	2.61842
3.81336	7.52652
0.00000	0.29955
−4.91377	−5.46357
8.11467	0.75415
−4.96210	−3.05211
8.11467	0.75415

Data were normalized in order to compare different models properly. Figure 1 shows the first 300 records in the normalized version of the dataset.



**Figure 1.** First 300 records of the normalized IBM stock and the S&P 500 index dataset.

### 3. Proposed Model

#### 3.1. Architecture and Inference

The proposed model is a multi-output generalization of the model introduced in Vlasenko et al. [33,34], based on the classical ANFIS [18] model. It consists of five layers that are shown in the Figure 2. The main difference with the architecture described in Vlasenko et al. [33] is that we have  $m$  outputs  $\hat{y}(k) = (\hat{y}_1(k), \hat{y}_2(k), \dots, \hat{y}_m(k))^T$  instead of just one  $\hat{y}$  for each input pattern  $x(k) = (x_1(k), x_2(k), \dots, x_n(k))^T$  and, correspondingly,  $m$  composite consequent layer units  $F = (F_1(k), F_2(k), \dots, F_m(k))^T$ . Figure 2 depicts the modified architecture.

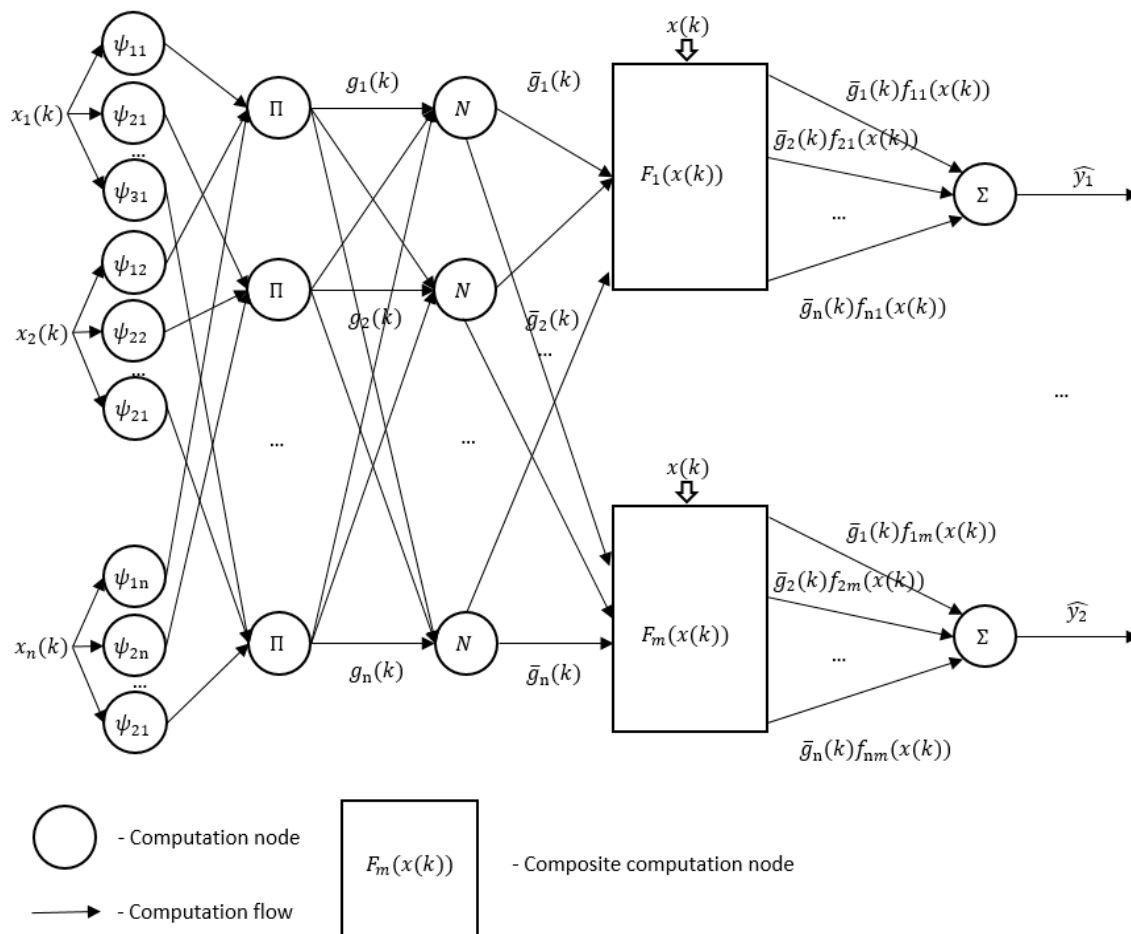


Figure 2. Architecture of the proposed neuro-fuzzy model.

The fuzzification layer consists of  $h^\psi$  membership functions for each input variable; hence, their total amount is  $h^\psi \times n$ . Gaussian membership functions are used—this type of membership function is also known as a radial basis function kernel, popular among a variety of machine learning techniques. It has the following form:

$$\psi_{jl}(x(k)) = \exp\left(-\frac{(x(k) - c_{jl}^\psi)^2}{2\sigma_{je}^2}\right), \tag{1}$$

where  $x(k)$  corresponds to a current input vector,  $c_{jl}^\psi$  is the centre, and  $\sigma_{jl}$  is a variety or width parameter, which controls the width of the bell curve.

On the initialization step, the first layer membership function centers  $c_{jl}^\psi$  are placed at equidistant widths fixed at  $\sigma_{jl} = 0.15$ . An example with  $h^\psi = 5$  is shown in Figure 3.

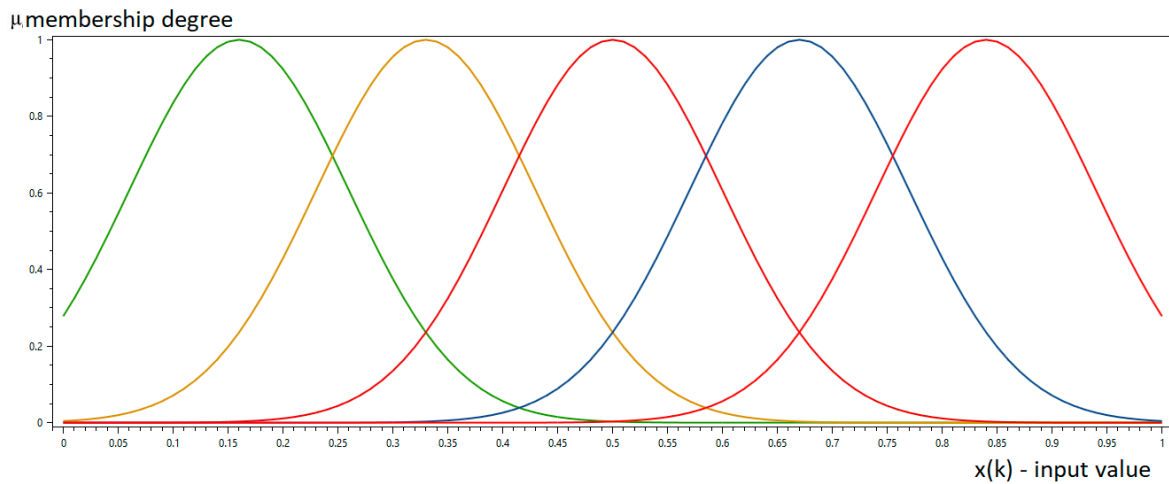


Figure 3. Example of the first layer membership functions.

The second layer performs an aggregation of the antecedent premises. It comprises  $h^\psi$  algebraic product fuzzy T-norm units  $g$ . Its outputs are computed by the formula:

$$g_j(k) = \prod_{i=1}^n \psi_{ji}(x_i(k)). \tag{2}$$

The third layer is non-parametrised and responsible for normalization. It also has  $h^\psi$  units, which outputs calculated by the following formula:

$$\bar{g}_j(k) = \frac{g_j(k)}{\sum_{j=1}^{h^\psi} g_j(k)} = \frac{\prod_{i=1}^n \psi_{ji}(x_i(k))}{\sum_{i=1}^{h^\psi} \prod_{i=1}^n \psi_{ji}(x_i(k))}. \tag{3}$$

This fulfils the Ruspini condition, which states the sum of various membership grades of one element is one [5]:

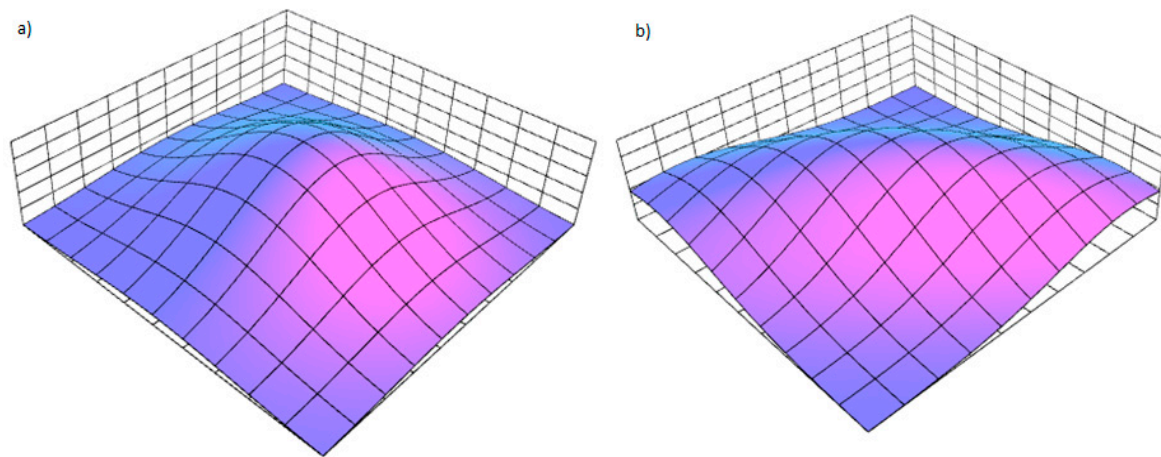
$$\sum_{j=1}^{h^\psi} \bar{g}_j(k) = 1. \tag{4}$$

The fourth layer is represented by the consequent units  $F_a$  that contain multidimensional Gaussian consequent functions  $\phi_{aje}(x(k))$  and weights  $p_{aje}$ . Multidimensional Gaussian functions replace the standard ANFIS polynomials and have the following form:

$$\phi_{aje}(x(k)) = \exp\left(-\frac{(x(k) - c_{aje}^\phi)^T Q_{aje}^{-1} (x(k) - c_{aje}^\phi)}{2}\right), \tag{5}$$

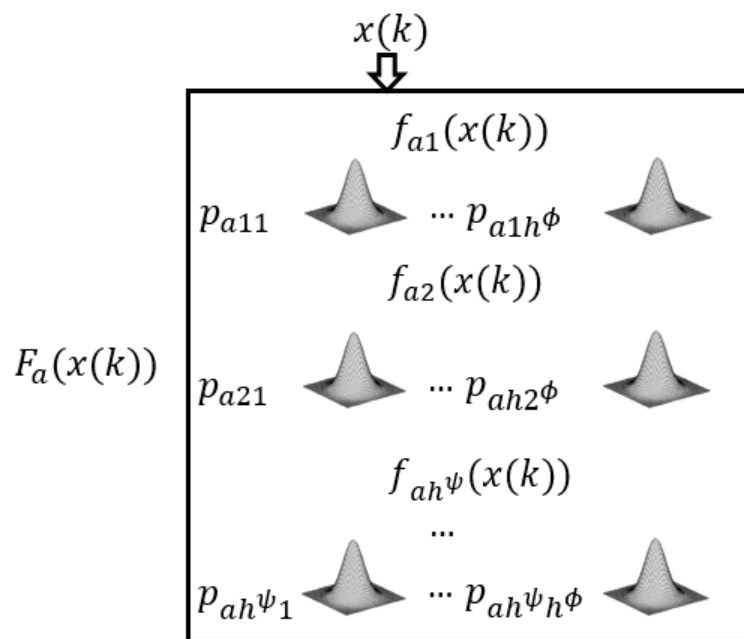
where  $x(k)$  represents an input vector,  $c_{aje}^\phi$  is a vector-centre of the current Gaussian, and  $Q_{aje}$  is the receptive field (covariance) matrix.

This function allows handling data distributed unevenly on the main axes. In case of a diagonal matrix, multidimensional Gaussian simply represents a collection of independent Gaussians with fixed width (Figure 4a).



**Figure 4.** The examples of multidimensional Gaussian with the same centre but different covariance matrices: (a) with diagonal matrix and (b) with matrix  $\begin{bmatrix} 0.9 & 0.6 \\ 0.7 & 0.9 \end{bmatrix}$

The detailed structure of the consequent unit  $F_a$ , for each output  $\hat{y}_a$  is depicted in Figure 5.



**Figure 5.** Detailed fourth layer representation.

The consequent layer output is:

$$f_{aj}(x(k)) = \sum_{e=1}^{h^\phi} p_{aje} \phi_{aje}(x(k)), \tag{6}$$

where  $h^\phi$  is an amount of multidimensional Gaussian functions for each unit  $f_j$ .

The output layer is non-parametrized and computed as a sum of its inputs:

$$\hat{y}_a = \sum_{j=1}^{h^\phi} g_{aj} f_{aj}(x(k)). \tag{7}$$

### 3.2. Learning Procedure

The proposed learning method generalizes the learning approach proposed by Vlasenko et al. [33] for a multivariate case. It comprises two simultaneous processes: optimization in the weights space  $P$  and in the consequent functions space.

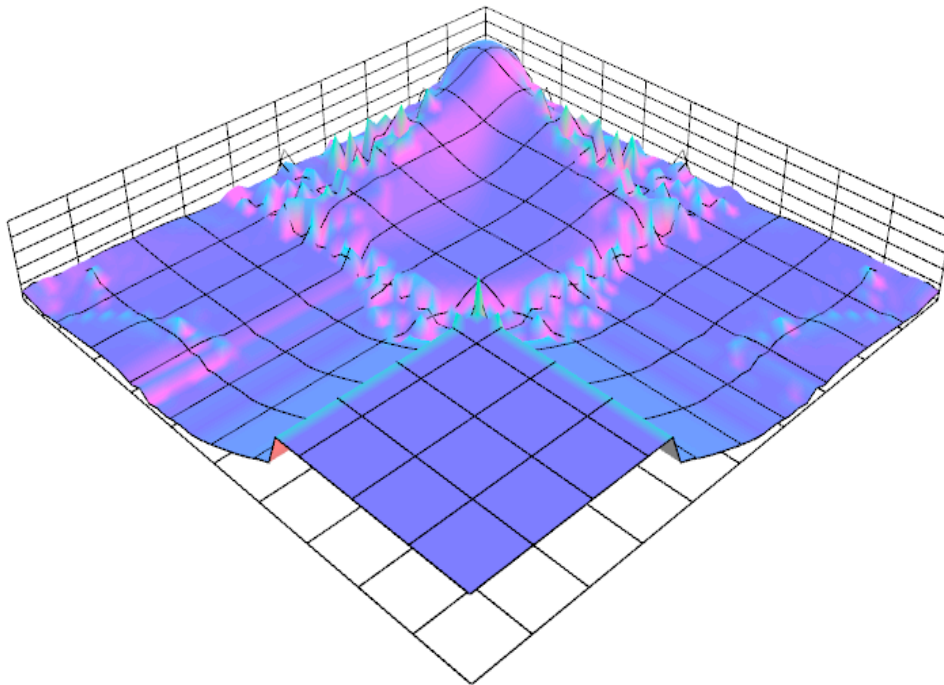
During the initialization step, the fourth layer function centres  $c_{ajl}^\phi$  are placed equidistantly and  $Q_{ajl}$  is initialized as identity matrices. Weights  $p$  are all originally set to 0.1.

Weights learning is performed by the Kachmarz method, which is a form of alternating projection method based on solving linear equation systems. It has gained popularity in signal processing, machine learning, and other domains. Each  $p_a$  has the following form:

$$p_a(k+1) = p_a(k) + \frac{y_a(k) - p_a^T f_a(x(k))}{f_a^T(x(k)) f_a(x(k))} f_a(x(k)), \quad (8)$$

where  $p_a(k)$  is a weights matrix,  $y_a(k)$  is a reference signal, and  $p_a^T f_a(x(k))$  is the model output.

Figure 6 shows a simplified example of the error surface in the weights space.



**Figure 6.** Weights error surface example.

The first-order stochastic, or online in terms of machine learning theory, gradient optimization method is used for consequent functions tuning. Stochastic gradient methods have been shown to have many advantages in comparison to popular batch gradient methods in real-life optimization problems including machine learning applications [35,36]. Instead of computing the full gradient over all dataset in batches, these methods update free model parameters after processing each input pattern. Their main advantage is the inherent noise of calculated gradients. Additionally, they do not suffer from rounding errors, which occur when having to store the accumulated gradient [37], and have better computational performance. The disadvantage is that they cannot be as easily parallelized as batch methods [36,37] and the severity of such restriction depends on the domain.

Learning is based on the standard mean square error criterion:

$$E = \frac{1}{N} \sum_{k=1}^N (y(k) - \hat{y}(k))^2, \quad (9)$$

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

The centres  $c_{ajl}^\phi$  are then tuned using:

$$\begin{cases} c_{ajl}^\phi(k+1) = c_{ajl}^\phi(k) + \lambda_c \frac{\tau_{ajl}^c(k)e(k)}{\eta_c(k)} \\ \eta_{ac}(k+1) = \beta_c \eta_{ac}(k) + \tau_{ajl}^c T \tau_{ajl}^c \end{cases}, \quad (10)$$

where  $\lambda_c$  is a learning step,  $\beta_c$  is a dumping parameter, and  $\tau_{ajl}^c$  is a vector of back propagated gradient values with respect to  $c_{ajl}^\phi$ .

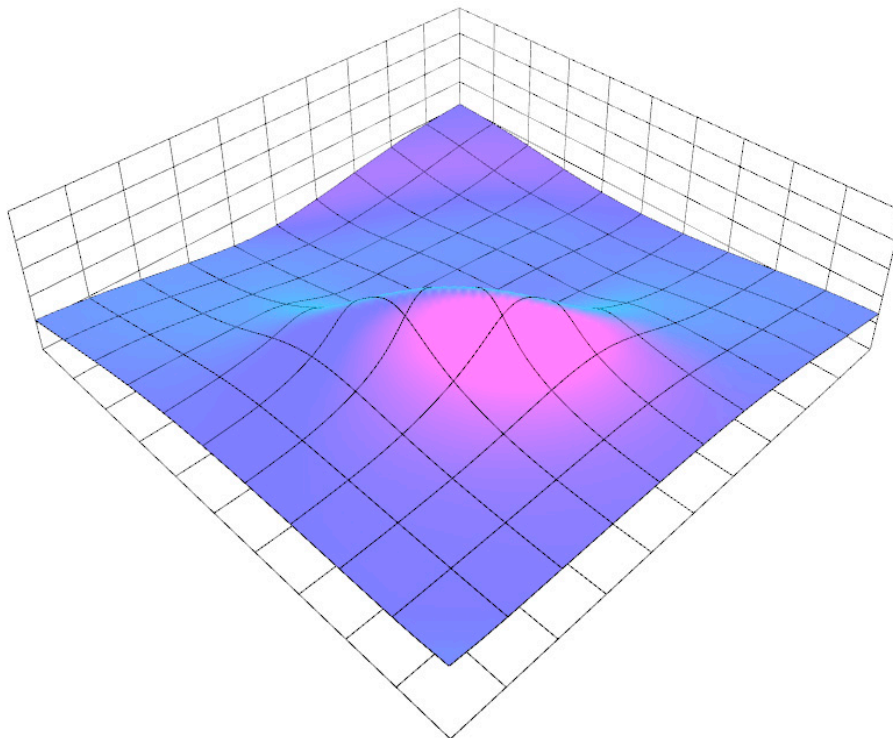
The  $Q_{ajl}$  matrices learning can be written as:

$$\begin{cases} Q_{ajl}(k+1) = Q_{ajl}(k) + \lambda_Q \frac{\tau_{ajl}^Q(k)e(k)}{\eta_{aq}(k)} \\ \eta_{aQ}(k+1) = \beta_Q \eta_{aQ}(k) + \text{Tr}(\tau_{ajl}^Q T \tau_{ajl}^Q) \end{cases}, \quad (11)$$

where  $\lambda_Q$  is a learning step,  $\beta_Q$  is a dumping parameter, and  $\tau_{ajl}^Q$  is a matrix of values back propagated with respect to  $Q_{ajl}$ .

Vectors  $\eta_{ac}$  and matrices  $\eta_{aQ}$  represent the decaying average value of past gradients and the algorithm initializes with  $\eta_{ac} = \eta_{aQ} = 10,000$ .

Figure 7 depicts an example of the error surface with respect to centres  $c_{ajl}^\phi$  and Figure 8 shows examples of multidimensional Gaussians after learning is finished.



**Figure 7.** An example of the error function surface in a simplified two-dimensional case.



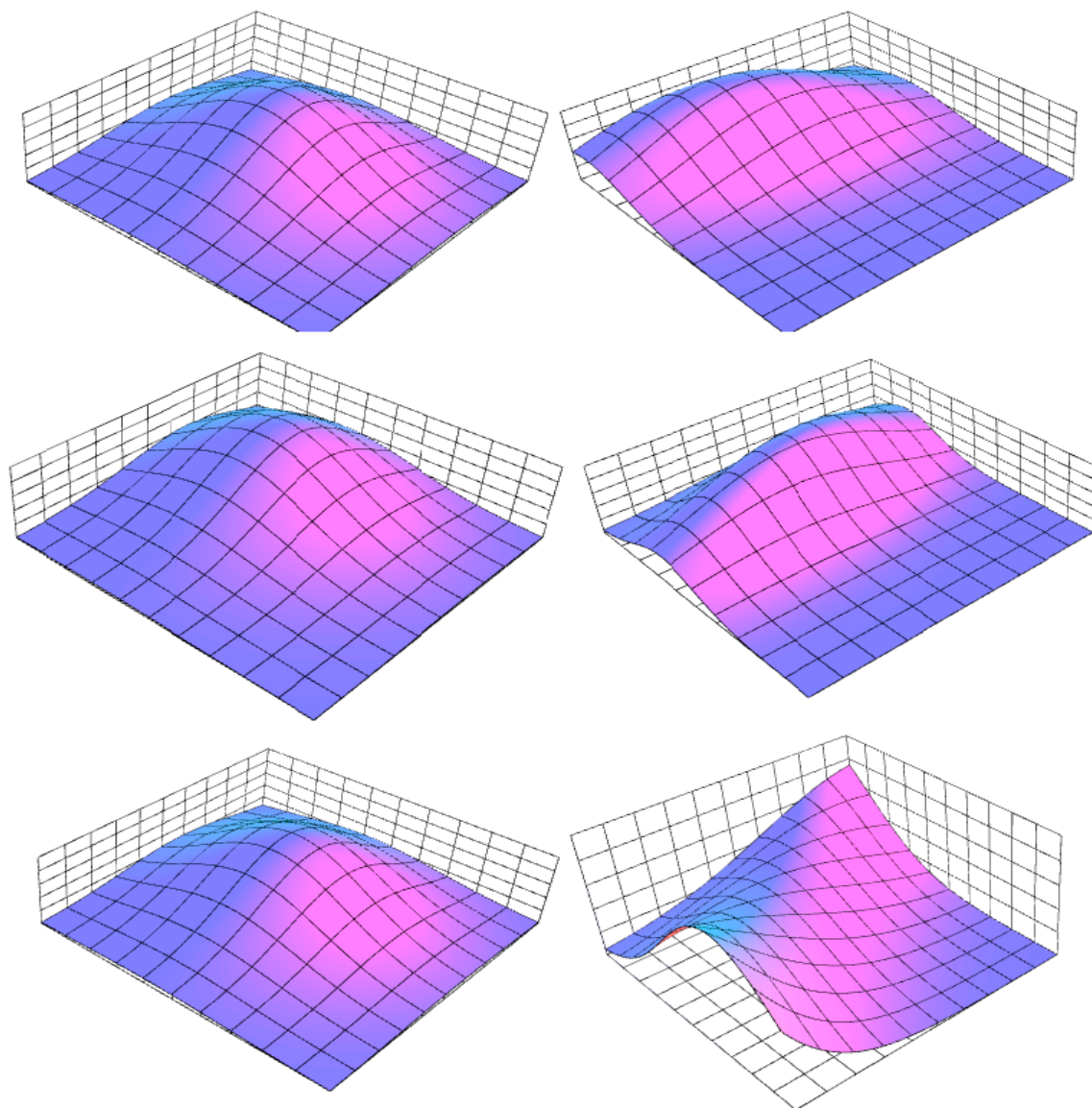


Figure 8. Examples of the two-dimensional Gaussian functions after learning.

#### 4. Experimental Results

We compared the prediction accuracy and computational performance to ANN using bipolar sigmoid activation functions. To train the ANN model, we used the Levenberg-Marquardt algorithm [38] and resilient backpropagation learning algorithm (RPROP) [39] as the popular batch optimization techniques.

The Accord.NET package [40] was used for the ANN implementations and we wrote custom software for the neuro-fuzzy model. In addition, we utilized the Math.NET Numerics package [41] for the linear algebra operations in the neuro-fuzzy model implementation.

The experiments were performed on a computer with Intel® Core(TM) Core i7-7700 processor (Intel, Santa Clara, California, U.S.) and 32 GB of memory.

Root Mean Square Error (*RMSE*) and Symmetric Mean Absolute Percent Error (*SMAPE*) criteria were used to estimate prediction accuracy:

$$RMSE = \sqrt{\frac{\sum_{k=1}^N (y(k) - \hat{y}(k))^2}{N}},$$

$$SMAPE = \frac{2}{N} \sum_{k=1}^N \frac{|\hat{y}(k) - y(k)|}{\hat{y}(k) + y(k)},$$

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

The bipolar sigmoid networks were built from sigmoids with an alpha value 0.4 for all cases, learning rate 0.67, and 50 epochs were used for resilient backpropagation. The Levenberg-Marquardt implementation was trained with 10 epochs. The neuro-fuzzy model had the following parameters  $h^\psi = 3$ ,  $\lambda_c = 1$ , and  $\lambda_Q = 1$ .

The original dataset was divided into a training set of 660 records and 228 records in the validation set. The input vectors for the bivariate case were composed of an equal number of sequential numbers from both dataset components.

We also compared the performance of all models in a univariate case, which is reflected in Table 2.

**Table 2.** Experimental results.

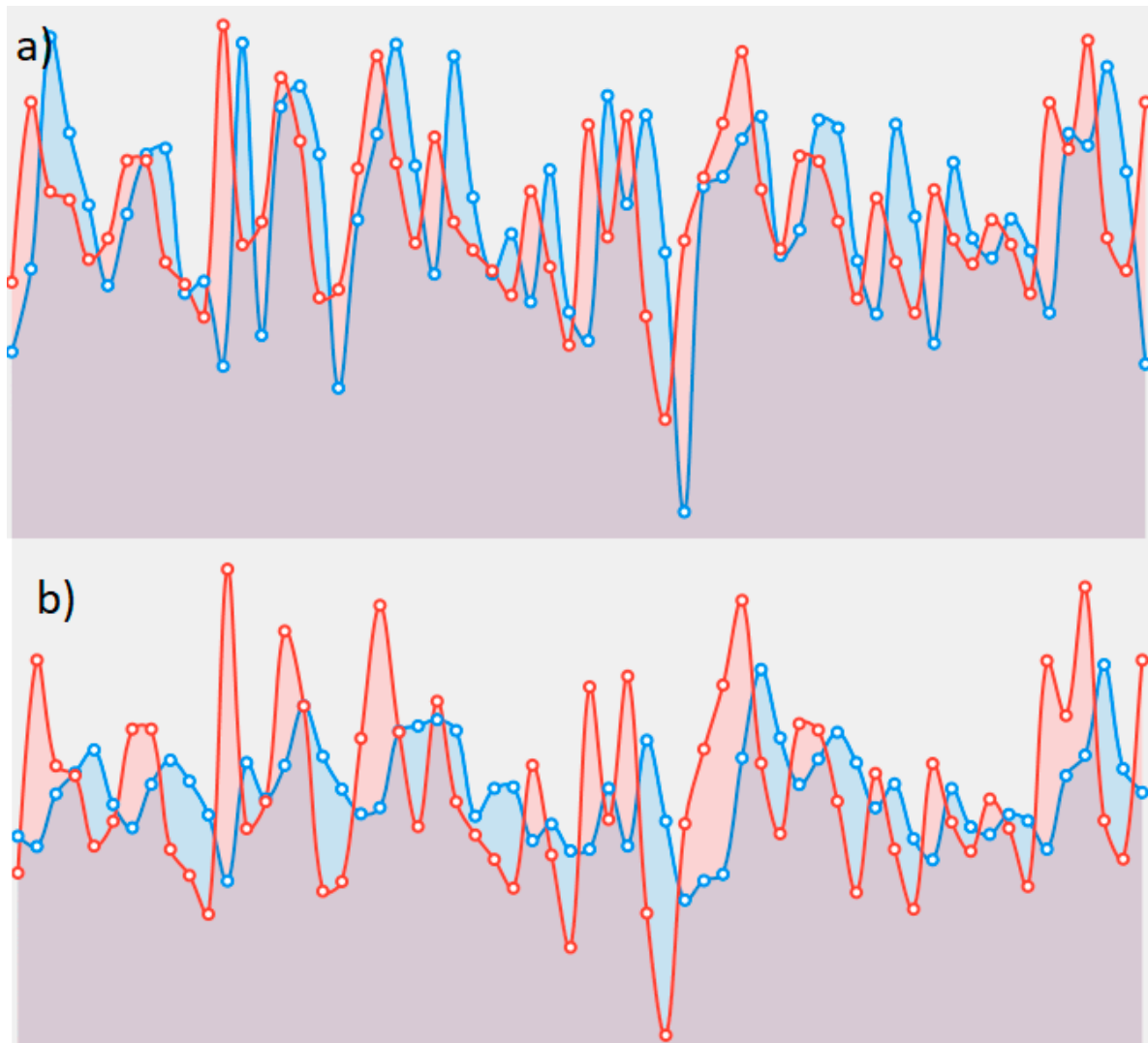
Model	IBM Stock Daily Log Returns and S&P 500 Index Dataset Results				
	Execution time (ms)	IBM stock RMSE (%)	IBM stock SMAPE (%)	S&P 500 index RMSE (%)	S&P 500 index SMAPE (%)
Proposed model $h^\psi = 1, \beta_c = 0.89, \beta_Q = 0.98$	64	8.938	10.049	4.443	4.594
$h^\psi = 1, \beta_c = 1.19, \beta_Q = 0.97$	39	12.956	20.548	-	-
$h^\psi = 1, \beta_c = 0.88, \beta_Q = 0.93$	38	-	-	6.097	9.005
Proposed model $h^\psi = 2, \beta_c = 0.85, \beta_Q = 0.93$	120	8.878	10.104	4.356	4.532
$h^\psi = 2, \beta_c = 0.82, \beta_Q = 0.91$	63	12.865	20.138	-	-
$h^\psi = 2, \beta_c = 0.75, \beta_Q = 0.92$	65	-	-	6.056	8.677
Proposed model $h^\psi = 4, \beta_c = 0.78, \beta_Q = 0.91$	257	8.866	10.013	4.294	4.463
$h^\psi = 4, \beta_c = 0.75, \beta_Q = 0.91$	126	12.693	20.133	-	-
$h^\psi = 4, \beta_c = 0.74, \beta_Q = 0.93$	110	-	-	6.113	8.775
Bipolar Sigmoid Network RBPR	847	8.91	10.156	4.66	4.96
	514	13.162	20.876		
	545	-	-	6.308	9.139
Bipolar Sigmoid Network Levenberg-Marquart	302	8.940	10.07	4.542	4.755
	313	14.544	22.664	-	-
	297	-	-	7.025	11.019

The performed experiments showed that ANN with a resilient backpropagation needs a significant number of epochs to attain good accuracy, greatly reducing computational performance. Levenberg-Marquardt learning requires fewer epochs, but each epoch costs more in terms of computing.

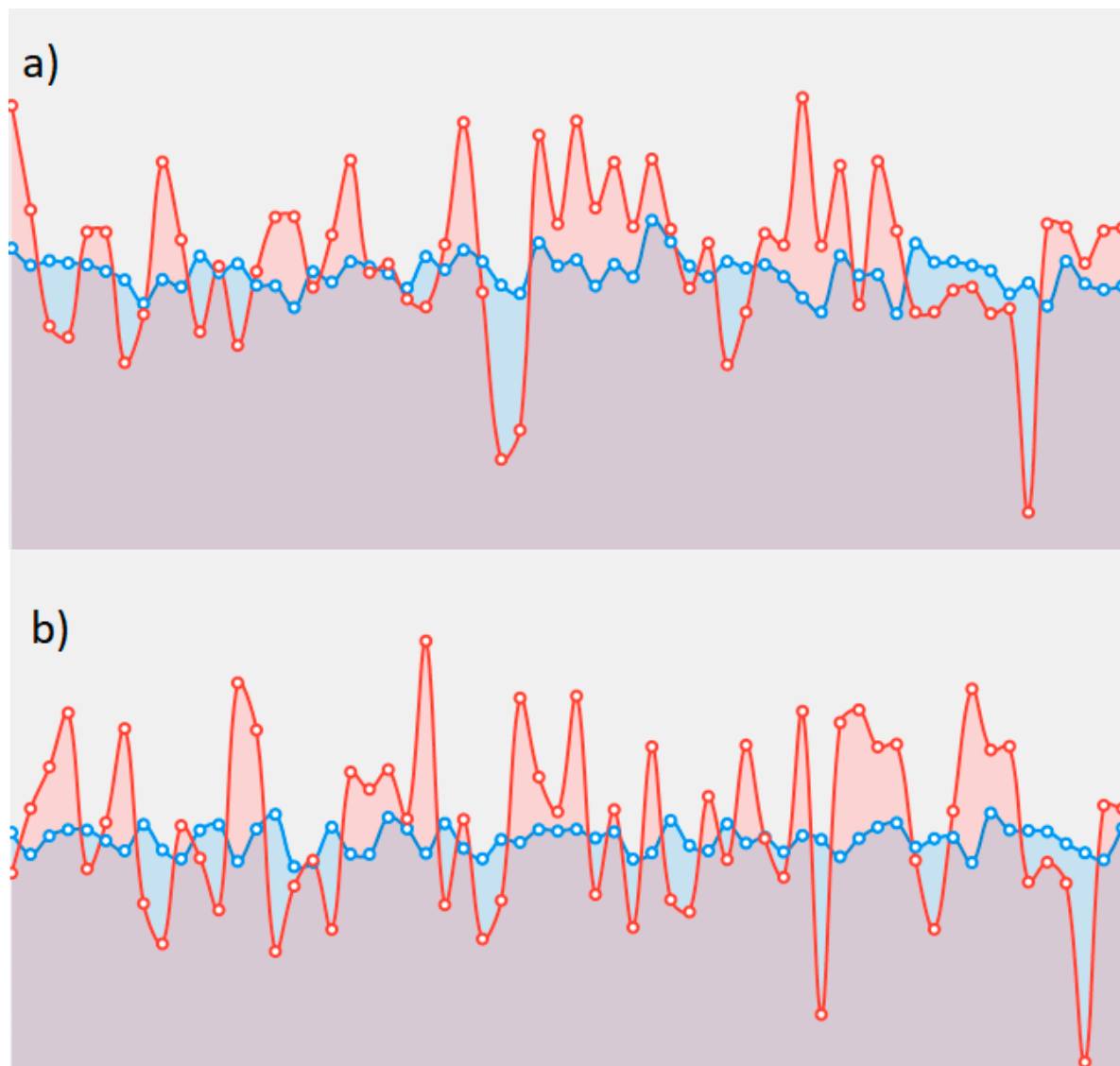
The proposed model demonstrated better performance results in all cases—both computational time and accuracy were significantly better than those of competitors. Also, the results in Table 2 show that accuracy improves with a larger  $h^\phi$ , but the downside is longer execution time.

The multivariate neuro-fuzzy model requires less computational resources than the two independent univariate models and has drastically better accuracy.

Visualization of the learning process is presented in Figures 9 and 10; IBM stock returns are depicted for the multivariate and univariate cases. Figure 9 shows the significant improvement in the forecasting plot obtained using the multivariate neuro-fuzzy model. Conversely, Figure 10 does not show such a degree of improvement for the bipolar sigmoid artificial neural model.



**Figure 9.** Learning process plot for the proposed neuro-fuzzy model: red dots—IBM stock returns actual values, blue—predicted values. (a) bivariate model; (b) univariate model.



**Figure 10.** Learning process plot for bipolar sigmoid artificial neural model with resilient backpropagation learning: red dots—IBM stock returns actual values, blue—predicted values. (a) bivariate model; (b) univariate model.

## 5. Conclusions

In this paper, we introduced a novel multiple output neuro-fuzzy model with multidimensional Gaussian functions in the consequent layer. The proposed model expands upon existing single output neuro-fuzzy models by adding separate fourth layer composed of computational units for each output. The model allows optimal training performance with respect to other outputs by implicitly handling lagged cross-correlation relations, which are common in financial data.

Experimental simulations demonstrated the good computational performance and prediction accuracy of our model in comparison to well-known ANN models. In addition, the advantage of utilizing multivariate time-series was proven.

**Author Contributions:** Conceptualization, O.V.; methodology, O.V. and D.P.; software, A.V.; validation, A.V. and N.V.; formal analysis, O.V.; investigation, A.V.; writing—original draft preparation, A.V.; writing—review and editing O.V. and N.V.; visualization, A.V. and N.V.; supervision, D.P.

**Funding:** This research received no external funding.

**Acknowledgments:** The authors thank the organizers of the DSMP'2018 conference for the opportunity to publish the article, as well as reviewers for the relevant comments that helped to better present the paper's material.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Commandeur, J.J.; Koopman, S.J. *An Introduction to State Space Time Series Analysis*; Oxford University Press: Oxford, UK, 2007.
2. Logue, A.C. The Efficient Market Hypothesis and Its Critics. *CFA Digest* **2003**, *33*, 40–41. [[CrossRef](#)]
3. Weigend, A.S.; Huberman, B.A.; Rumelhart, D.E. Predicting the future: A connectionist approach. *Int. J. Neural Syst.* **1990**, *1*, 193–209. [[CrossRef](#)]
4. Crone, S.F.; Hibon, M.; Nikolopoulos, K. Advances in forecasting with neural networks? Empirical evidence from the NN3 competition on time series prediction. *Int. J. Forecast.* **2011**, *27*, 635–660. [[CrossRef](#)]
5. Bodyanskiy, Y.; Popov, S. Neural network approach to forecasting of quasiperiodic financial time series. *Eur. J. Oper. Res.* **2006**, *175*, 1357–1366. [[CrossRef](#)]
6. Dubois, D.; Prade, H. The legacy of 50 years of fuzzy sets: A discussion. *Fuzzy Sets Syst.* **2015**, *281*, 21–31. [[CrossRef](#)]
7. Dubois, D.; Prade, H. Gradualness, uncertainty and bipolarity: Making sense of fuzzy sets. *Fuzzy Sets Syst.* **2012**, *192*, 3–24. [[CrossRef](#)]
8. Sheta, A. Software Effort Estimation and Stock Market Prediction Using Takagi-Sugeno Fuzzy Models. In Proceedings of the 2006 IEEE International Conference on Fuzzy Systems, Vancouver, BC, Canada, 16–21 July 2006. [[CrossRef](#)]
9. Chang, P.-C.; Liu, C.-H. A TSK type fuzzy rule based system for stock price prediction. *Expert Syst. Appl.* **2008**, *34*, 135–144. [[CrossRef](#)]
10. Pamučar, D.; Cirovic, G. Vehicle Route Selection with an Adaptive Neuro Fuzzy Inference System in Uncertainty Conditions. *Decis. Mak. Appl. Manag. Eng.* **2018**, *1*, 13–37. [[CrossRef](#)]
11. Pamučar, D.; Ljubojević, S.; Kostadinović, D.; Đorović, B. Cost and risk aggregation in multi-objective route planning for hazardous materials transportation—A neuro-fuzzy and artificial bee colony approach. *Expert Syst. Appl.* **2016**, *65*, 1–15. [[CrossRef](#)]
12. Bodyanskiy, Y.; Popov, S.; Rybalchenko, T. Multilayer Neuro-fuzzy Network for Short Term Electric Load Forecasting. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 339–348.
13. Otto, P.; Bodyanskiy, Y.; Kolodyazhnyi, V. A new learning algorithm for a forecasting neuro-fuzzy network. *Integr. Comput.-Aided Eng.* **2003**, *10*, 399–409. [[CrossRef](#)]
14. Sremac, S.; Tanackov, I.; Kopic, M.; Radović, D. ANFIS model for determining the economic order quantity. *Decis. Mak. Appl. Manag. Eng.* **2018**, *1*. [[CrossRef](#)]
15. De Jesús Rubio, J. USNFIS: Uniform stable neuro fuzzy inference system. *Neurocomputing* **2017**, *262*, 57–66. [[CrossRef](#)]
16. Lukovac, V.; Pamučar, D.; Popović, M.; Đorović, B. Portfolio model for analyzing human resources: An approach based on neuro-fuzzy modeling and the simulated annealing algorithm. *Expert Syst. Appl.* **2017**, *90*, 318–331. [[CrossRef](#)]
17. Kar, S.; Das, S.; Ghosh, P.K. Applications of neuro fuzzy systems: A brief review and future outline. *Appl. Soft Comput.* **2014**, *15*, 243–259. [[CrossRef](#)]
18. Jang, J.S. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Trans. Syst. Man Cybern.* **1993**, *23*, 665–685. [[CrossRef](#)]
19. Mendes, J.; Souza, F.; Araújo, R.; Rastegar, S. Neo-fuzzy neuron learning using backfitting algorithm. *Neural Comput. Appl.* **2017**. [[CrossRef](#)]
20. Silva, A.M.; Caminhas, W.; Lemos, A.; Gomide, F. A fast learning algorithm for evolving neo-fuzzy neuron. *Appl. Soft Comput.* **2014**, *14*, 194–209. [[CrossRef](#)]
21. Bodyanskiy, Y.V.; Tyshchenko, O.K.; Kopaliani, D.S. Adaptive learning of an evolving cascade neo-fuzzy system in data stream mining tasks. *Evol. Syst.* **2016**, *7*, 107–116. [[CrossRef](#)]
22. Billah, M.; Waheed, S.; Hanifa, A. Stock market prediction using an improved training algorithm of neural network. In Proceedings of the 2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE), Rajshahi, Bangladesh, 8–10 December 2016. [[CrossRef](#)]

23. Esfahanipour, A.; Aghamiri, W. Adapted Neuro-Fuzzy Inference System on indirect approach TSK fuzzy rule base for stock market analysis. *Expert Syst. Appl.* **2010**, *37*, 4742–4748. [[CrossRef](#)]
24. Rajab, S.; Sharma, V. An interpretable neuro-fuzzy approach to stock price forecasting. *Soft Comput.* **2017**. [[CrossRef](#)]
25. Hadavandi, E.; Shavandi, H.; Ghanbari, A. Integration of genetic fuzzy systems and artificial neural networks for stock price forecasting. *Knowl.-Based Syst.* **2010**, *23*, 800–808. [[CrossRef](#)]
26. Bodyanskiy, Y.; Pliss, I.; Vynokurova, O. Adaptive wavelet-neuro-fuzzy network in the forecasting and emulation tasks. *Int. J. Inf. Theory Appl.* **2008**, *15*, 47–55.
27. Chiu, D.-Y.; Chen, P.-J. Dynamically exploring internal mechanism of stock market by fuzzy-based support vector machines with high dimension input space and genetic algorithm. *Expert Syst. Appl.* **2009**, *36*, 1240–1248. [[CrossRef](#)]
28. Parida, A.K.; Bisoi, R.; Dash, P.K.; Mishra, S. Times Series Forecasting using Chebyshev Functions based Locally Recurrent neuro-Fuzzy Information System. *Int. J. Comput. Intell. Syst.* **2017**, *10*, 375. [[CrossRef](#)]
29. Atsalakis, G.S.; Valavanis, K.P. Forecasting stock market short-term trends using a neuro-fuzzy based methodology. *Expert Syst. Appl.* **2009**, *36*, 10696–10707. [[CrossRef](#)]
30. Ebadzadeh, M.M.; Salimi-Badr, A. CFNN: Correlated fuzzy neural network. *Neurocomputing* **2015**, *148*, 430–444. [[CrossRef](#)]
31. Tsay, R.S. *Analysis of Financial Time Series*; Wiley Series in Probability and Statistics; John Wiley & Sons: Hoboken, NJ, USA, 2005. [[CrossRef](#)]
32. Monthly Log Returns of IBM Stock and the S&P 500 Index Dataset. Available online: <https://faculty.chicagobooth.edu/ruey.tsay/teaching/fts/m-ibmspln.dat> (accessed on 1 September 2018).
33. Vlasenko, A.; Vynokurova, O.; Vlasenko, N.; Peleshko, M. A Hybrid Neuro-Fuzzy Model for Stock Market Time-Series Prediction. In Proceedings of the IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 21–25 August 2018. [[CrossRef](#)]
34. Vlasenko, A.; Vlasenko, N.; Vynokurova, O.; Bodyanskiy, Y. An Enhancement of a Learning Procedure in Neuro-Fuzzy Model. In Proceedings of the IEEE First International Conference on System Analysis & Intelligent Computing (SAIC), Kyiv, Ukraine, 8–12 October 2018. [[CrossRef](#)]
35. LeCun, Y.A.; Bottou, L.; Orr, G.B.; Müller, K.-R. Efficient BackProp. *Neural Netw. Tricks Trade* **2012**, 9–48. [[CrossRef](#)]
36. Bottou, L.; Curtis, F.E.; Nocedal, J. Optimization Methods for Large-Scale Machine Learning. *SIAM Rev.* **2018**, *60*, 223–311. [[CrossRef](#)]
37. Wiesler, S.; Richard, A.; Schluter, R.; Ney, H. A critical evaluation of stochastic algorithms for convex optimization. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–30 May 2013. [[CrossRef](#)]
38. Kanzow, C.; Yamashita, N.; Fukushima, M. Erratum to “Levenberg–Marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints. *J. Comput. Appl. Math.* **2005**, *177*, 241. [[CrossRef](#)]
39. Riedmiller, M.; Braun, H. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In Proceedings of the IEEE International Conference on Neural Networks, Rio, Brazil, 8–13 July 2018. [[CrossRef](#)]
40. Souza, C.R. The Accord.NET Framework. Available online: <http://accord-framework.net> (accessed on 1 November 2018).
41. Math.NET Numerics. Available online: <https://numerics.mathdotnet.com> (accessed on 1 November 2018).

