DOI 10.29042/2019-5615-5620

Helix Vol. 9 (5): 5615- 5620

Neural Network Architecture Development for Time Series Forecasting

[1]Alexander Zakharov, [2]Galim Vakhitov, [3]Zulfira Enikeeva
[1-3] Kazan Federal University
Email: GZVahitov@kpfu.ru

Received: 20th August 2019, Accepted: 30th September 2019, Published: 31st October 2019

Abstract
The paper reveals the results of studies on the development of neural network architecture for time series forecasting. Three types of neural networks were investigated: radial network, convolutional network, and multilayer perceptron. The main objective of this study is to identify, based on a comparative analysis, the architecture of a neural network, which is most suitable for the specifics of forecasting data of a financial nature. In the course of work, from the more than 1000 configurations of neural networks, the most successful ones were selected, with the smallest standard error. The C# programming language and the Visual Studio development environment were chosen in the capacity of the implementation tools. In the work, the back propagation of error algorithm was used as a method for training the neural network. During the training of the network, a dynamic change in the network configuration was carried out. The time series of the financial market were used as the data supplied to the neural network. The work was carried out on the data of exchange rates for 2000-2018. As a result of the study, it was determined that the use of a multilayer perceptron provides higher accuracy. The results of the work can be applied to forecast the values of securities, currency pairs and other financial instruments.

Keywords
Time Series, Neural Network, Radial Neural Network, Convolutional Neural Network, Forecasting, Neural Network Architecture.

Introduction

The analysis of time series by machine learning methods has received considerable attention [1], [4], [5]. For this, such methods as Multi-Scale Methods (convolutional neural networks), Time-series Shapelets (Wang, Oates) [2], Time Warping Symbolic Aggregation Approximation (L. Ye and E. Keogh) [3] and radial, convolutional, and multilayer neural networks architectures are used. Also, the research results were highlighted in such works as [6], [7], [8], [9]. Our work can partly be considered as a continuation of [10]; therefore, we will consider data on changes in exchange rates as time series. The main goal of the work is to identify a neural network configuration that is most suitable for forecasting the time series values based on studying the mean square error (MSE) value for a neural network with the architecture developed by us. The work was carried out proceeding from the data on exchange rates for 2000-2018.

Methods

When identifying the neural network architecture which is most effective in forecasting time series, we will use models of neurons of three types:
- input neurons which values are set from the outside and which do not have ancestors;
- output neurons which values are used externally and which, as a rule, have no descendants;
- internal neurons that do not belong to either the first or second class.

The model of a neuron is presented below in Figure 1. With this structure, the neuron collects with some weights the values of the ancestors, summarizes them, and then applies a certain nonlinear function to the sum. The following are usually nonlinear functions :

1) Sigmoid. $$S(x) = \frac{1}{1 + e^{-x}} \quad (1)$$

2) Tanh.S (x) = $$\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (2)$$

3) ReLu.S (x) = Max (0, x) (3)

In addition, in many architectures of modern neural networks, neurons form layers that are ordered. Inside one layer, the neurons are not interconnected, there are connections only with neighbouring layers - the inputs are taken from the previous layer, and the calculated value is transferred to the next one in order. The first layer consists of input neurons, so it is called the input. The latter is from the output neurons, so its name is the output. Similarly, all other layers are called inner.

We have a function y. In this case, the sigmoid function was used, therefore, $S(x) = \frac{1}{1+e^{-x}}$, (4)

Where y is the output signal of the neuron.

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

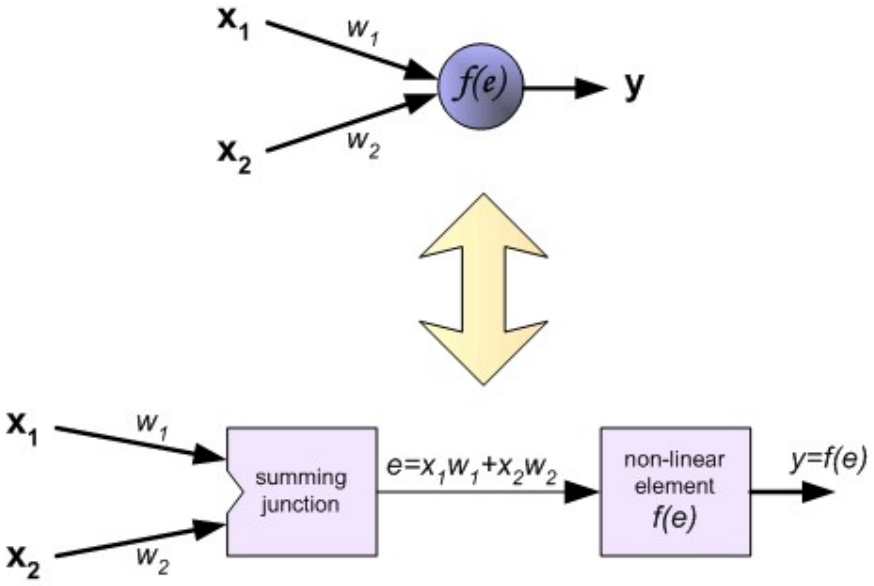

Fig. 1: Artificial Neuron.

In order to train a neural network, we need to prepare data for training (training set). In Fig.1, training sample data consists of input signals (x1 and x2) and the desired result z. Learning is a series of iterations. At each iteration the weights of the neurons are adjusted using new data from training examples. Changing weights is the essence of the algorithm described below.

Step 1. Receiving a signal at the output. Each learning step begins with exposure to input from training examples. After that, we can determine the values of the output signals for all neurons in each layer of the network.

Step 2. At this step of the algorithm, the network output signal **y is** compared with the desired output signal **z**, which is stored in the training data. The difference between these two signals is called the error **d of the** output layer of the network.

Step 3. Weight correction **W (X_m) n**. When the signal error value for each neuron is calculated, weights of each input node (dendrite) of the neuron can be adjusted. In the formulas below, df (e) / de is the derivative of the neuron activation function (which weights are adjusted).

Since the activation function is sigmoid, its derivative will be

$$S`(x) = S(x) * \big(1 - S(x)\big) \quad (5)$$

Using a radial neural network to predict time series

The construction of the radial network is performed in three layers. The first layer contains input neurons followed by radial and then output. It should be noted that in this work, all configurations of radial networks have the same number of neurons in the input and radial layers. The following is an example of a radial neural network.

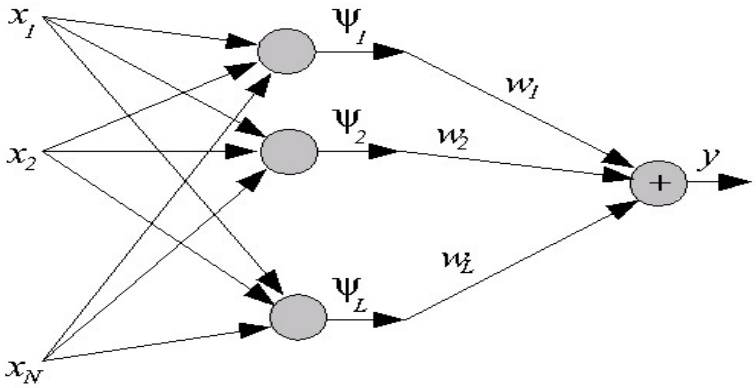

Fig. 2: Scheme of the Radial Neural Network.

Here $fi_i(|X\text{-}C_i|)$ is the activation function of the *i-th* radial neuron, which in the literature can be called the basic function. The network performs some data transformations. In particular, there is a constant "comparison" with the reference weights of neurons, which are the connections between the input and radial layers (to simplify the calculations, we put $w_0 = 0$):

$$y = sum\ [i = 1: L]\ (w_i * fi_i\ (|\ XC_i\ |)) \quad (6)$$

The task of training the network is to select such values of L, C_i and w_i that deliver a minimum of the objective function

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

$E = (1/2) * sum [k = 1: p] ((sum [i = 0: L] (w_i * fi (| XC_i |)) - d^k)^2)$ (7)

Where p is the number of training samples.

The training in our modification takes place according to the Back propagation of Errors algorithm described above.

Training sample preparation

As a training sample for neural networks, data from the website of the Central Bank of the Russian Federation https://www.cbr.ru/ will be used. Data was taken, starting in 2000, for different currencies, where there is data for each day (for example, the exchange rate of the euro, dollar, etc. relative to the rouble). The data at the input of the neural network are normalized (so that the values of the time series are within [0; 1]).

The initial weights for all types of neural networks that participate in this work will be randomly filled with data from the interval [0; 1].

The forecasting error will be calculated by the formula:

$$MSE = \sum \frac{(x_i - ouput_i)^2}{n}, \quad (8)$$

Where x_i are values obtained using a neural network; $ouput_i$ are the values that should be obtained at the output (values of the training sample); n is the number of output neurons, in this case there are 3 of them. With each iteration of network learning, an error should decrease.

For each type of neural network, we will receive several configurations, where the number of inputs, the number of intermediate layers, and the number of neurons in the intermediate layers will change. Next, we get a table with an error on the test data sets (datasets). The number of output neurons will always be fixed and equal to 3. A value of 1-0-0 will mean that an increase is predicted, a value of 0-1-0 means that lateral movement is predicted, a value of 0-0-1 will mean that a decrease in trend is predicted. To determine the nature of the trend, we will use the least squares method. After determining the coefficient when constructing a direct trend, y = a * x + b, we will take the following:

1) an uptrend, if the coefficient a> 0.05;

2) the trend does not change if the coefficient a> = - 0.05 and a <= 0.05;

3) a downward trend if the coefficient a <- 0.05.

The trend in the work will be compiled for 7 days, so the least squares method will be applied to 7 points from which the coefficient a will be calculated.

Additional modifications

In our work, we also tested a convolutional layer. The convolutional neural network will always have 24 inputs. Next, the data is converted over 24 days to a 24x24 matrix. The remaining rows are formed by randomly mixing 24 data. Next, we will run a 3 by 3 frame on this matrix and find the highest value in this frame. Thus, we get a total of 64 frames from a 24 by 24 matrix and find 64 elements. Next, these 64 elements are sent to the input of the multilayer perceptron, in which the number of neurons in the hidden layer will change.

Also in our work, we used a hybrid neural network learning algorithm. Radial neurons will summarize the input signals according to the formulas above, constantly comparing with reference weights. As a reference weights, we took the dollar exchange rate starting in 2000. Further, the received amount passes the activation function and transfers information to the output neurons. Then the error propagates back to the radial neurons and the correction of the weights between the layer of radial neurons and the output. It is worth noting that in our implementation, the reference weights between the layer of input neurons and radial ones will not be adjusted during the training process, which may affect the final result.

Unlike standard libraries, for example keras (python), in our implementation we can dynamically add neurons to the network configuration during training.

Results and Discussion

Below are tables for calculations of the mean square errors in neural networks with different architectures. The first column will be the average error of 50 test data. The dollar exchange rate for each day, starting in 2000, was taken as test data. The number of input neurons ranged from 6 to 30, the number of neurons in the middle (hidden) layer ranged from 6 to 49. The number of output neurons remained unchanged - 3.

In total, more than 1000 configurations of neural networks were obtained; the table shows the most efficient architectures that produce the smallest error.

The following is a calculation table for a multilayer perceptron.

No.	Error	The number of input neurons	The number of hidden layer neurons
1	0.050353	18	15
2	0.04417	18	16
3	0.05645	18	17
4	0.045013	18	18
5	0.041851	18	19
6	0.050322	18	20
7	0.058771	18	21

Table 1: MSE for Multi-Layer Perceptron Configurations

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

The smallest error was shown by a network with a configuration of 18 input neurons and 19 neurons of the hidden layer. With 18 input neurons, the MSE error rate does not vary much and fluctuates around 0.05. With other configurations not shown in the table, the MSE error indicator is either around 0.05 or worse.
The following is a calculation table for a radial neural network.

No.	Error	The number of input neurons
1	0.097814	12
2	0.033681	13
3	0.125782	14
4	0.08886	15
5	0.058433	16
6	0.113122	17
7	0.10263	18

Table 2: MSE for Radial Neural Network Configurations.

As we can see, the spread of MSE errors in this table is much larger than in other tables. The best performance is in a network with 13 inputs and 13 radial neurons. This variation is due to the reference weights that were assigned to the network before the training.
The following is a calculation table for a convolutional neural network.

No.	Error	The number of hidden layer neurons
1	0.074377	21
2	0.074376	22
3	0.074376	23
4	0.066376	24
5	0.074376	25
6	0.062267	26
7	0.074376	27

Table 3: MSE for Convolutional Neural Network Configurations

As can be seen from the table, the best indicator is for a network with 24 inputs and 26 neurons of the hidden layer. But, nevertheless, these results are worse than those of the best radial and multi-layer network configuration. This may be due to data conversion using the convolutional layer presented in this paper.
Below are tables with results of mean square error calculations for neural networks with different architectures for data on a number of shares of the Russian stock market. That is, the network that trained with the use of data on currencies was tested on data on stocks, which, by analogy with the trend, should be assigned to one of three classes. The number of neural network inputs is always 6. The number of neurons in the hidden layer changed. There were two test samples for shares.
The following tables have been obtained. Below are the tables for a multilayer perceptron.

No.	Error	The number of input neurons	The number of output neurons
1	0.206549	6	5
2	0.206211	6	6
3	0.206674	6	7
4	0.203641	6	8
5	0.203319	6	9
6	0.203335	6	10
7	0.206294	6	11

Table 4.1: MSE for Multilayer Perceptron Configurations

No.	Error	The number of input neurons	The number of output neurons
1	0.205303	6	5
2	0.216737	6	6
3	0.205742	6	7
4	0.220854	6	8
5	0.204751	6	9
6	0.216155	6	10
7	0.206494	6	11

Table 4.2: MSE for Multilayer Perceptron Configurations

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

Compared to the table, we observe deterioration in indicators, both for sample 1 and sample 2. This is due to the fact that the nature of data has changed, and the number of input neurons has decreased, which a priori implies deterioration in the result.
Since the amount of input in the radial network coincides with the number of radial neurons in the hidden layer, that is, only one is obtained. The average error value for sample 1 turned out to be 0.411752, and for the second sample - 0.41882, which is a rather poor indicator among all configurations of neural networks.
Below are the tables for the convolutional neural network.

No.	Error	The number of hidden layer neurons
1	0.20663	22
2	0.206629	23
3	0.203962	24
4	0.220221	25
5	0.203964	26
6	0.209861	27
7	0.20663	28

Table 5.1. MSE for Convolutional Neural Network Configurations

No.	Error	The number of hidden layer neurons
1	0.205677	15
2	0.231375	16
3	0.231042	17
4	0.21106	18
5	0.206563	19
6	0.207625	20
7	0.203491	21

Table 5.2: MSE for Convolutional Neural Network Configurations

These indicators also worsened compared to table 3. This is also due to a change in the nature of the data and the peculiarities of the change in the convolutional layer, since it is designed for 24 inputs, and not for 6.

Summary

As a result of the work, more than 1000 configurations of neural networks were obtained and tested. The best indicators for forecasting a trend for a week (increase, decrease or immutability) were shown by the configuration of a multilayer perceptron with the number of inputs from 18 to 22. In other words, to forecast a trend for a week, it is necessary to rely on data on currencies during the previous 18-22 days. According to these indicators, this configuration of neural networks gives the smallest error. In the case of using a radial neural network, the best performance was achieved using a configuration with 16 input and 16 output neurons. But this network still plays a little with the configurations of the multilayer perceptron described above. Regarding the convolutional neural network, it always used 24 inputs, the number of neurons in the hidden layer changed, but the indicators turned out to be worse than in the best versions of the multilayer perceptron and radial network, but better than the indicators of a multilayer perceptron with less than 18 inputs.
Thus, to forecast the trend in the movement of exchange rates for a week, it is necessary to rely on the readings for these currencies for the 16-22 previous days; in this case the neural network will give the smallest error and make the trend prediction as correct as possible.

Regarding networks trained on the basis of currency data, but tested on the basis of stock data, we see the following. The error rate for all types of networks has worsened in the range of 0.1-0.15, which is quite significant. Here, the convolution network gave us the best performance, slightly ahead of the configuration of the multilayer perceptron, which is observed in the tables. A radial neural network with 6 inputs and 6 radial neurons showed the worst result.

Conclusions

This study may be useful in the formation of a methodology for identifying the configuration of neural networks, allowing the forecast for the values of time series with maximum accuracy. As a result of the study, it was determined that the use of a multilayer perceptron provides higher accuracy. The network configurations presented in this paper can be used to forecast exchange rates, stocks and other financial instruments.

Acknowledgements

The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University.

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

References

[1] Van The Huy и Duong Tuan Anh. “An efficient implementation of anytime k-medoids clustering for time series under dynamic time warping”. B: Proceedings of the Seventh Symposium on Information and Communication Technology, SoICT 2016, Ho Chi Minh City, Vietnam, December 8-9, 2016. 2016, с. 22—29. doi: 10.1145/3011077. 3011128. url: http://doi.acm.org/10.1145/3011077.3011128.

[2] Zhiguang Wang и Tim Oates. “Time Warping Symbolic Aggregation Approximation with Bag-of-Patterns Representation for Time Series Classification”. B: 13th International Conference on Machine Learning and Applications, ICMLA 2014, Detroit, MI, USA, December 3-6, 2014. 2014, с. 270—275. doi: 10.1109/ICMLA.2014.49. url: http://dx.doi.org/10.1109/ICMLA.2014.49.

[3] L. Ye и E. Keogh. “Time series shapelets: a new primitive for data mining”. B: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2009.

[4] Jason Lines и др. “A shapelet transform for time series classification”. B: The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’12, Beijing, China, August 12-16, 2012. 2012, с. 289—297. doi: 10. 1145 / 2339530.2339579. url: http://doi.acm.org/10.1145/2339530.2339579.

[5] Martin Wistuba, Josif Grabocka и Lars Schmidt-Thieme. “Ultra-Fast Shapelets for Time Series Classification”. B: CoRR abs/1503.05018 (2015). url: http://arxiv. org/abs/1503.05018.

[6] Zhicheng Cui, Wenlin Chen и Yixin Chen. “Multi-Scale Convolutional Neural Networks for Time Series Classification”. B: CoRR abs/1603.06995 (2016). url: http: //arxiv.org/abs/1603.06995.

[7] Zhiguang Wang и Tim Oates. “Spatially Encoding Temporal Correlations to Classify Temporal Data Using Convolutional Neural Networks”. B: CoRR abs/1509.07481 (2015). url: http://arxiv.org/abs/1509.07481.

[8] Latypova R., Tumakov D. Method of selecting an optimal activation function in perceptron for recognition of simple objects // Proceedings of 16th IEEE East-West Design and Test Symposium, 2018, P. 390-394.

[9] Jiang J.-Q., Wei-Xing Zhou W.-X., Sornette D., Woodard R., Bastiaensen K., Cauwels P.Bubble Diagnosis and Prediction of the 2005-2007 and 2008- 2009 Chinese stock market bubbles, Journal of Economic Behavior & Organization 74 (3), 149-162 (2010).

[10] Vakhitov G.Z., Enikeeva Z.A. The use of neural networks to predict the dynamics of the stock market / Information Technology and Mathematical Modeling (ITMM-2017): Materials of the XVI Conference named after A.F. Terpugov (September 29 - October 3, 2017). - Tomsk: NTL Publishing House, 2017. - Part 2. - p. 264-267.

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495