Prospects of LSTM Neural Networks Use in Supply Chain Management When Developing a Crypto Currency Rate Forecast

Marat Rashitovich Safiullin [1], Leonid Alekseevich Elshin [2], Ayzat Minekhanovich Gilmanov [3]
[1,2,3] *Kazan Federal University*
[2] State Budgetary Institution "Centre of Perspective Economic Researches", Academy of Sciences, Republic of Tatarstan
[2] Kazan National University of Science and Technology
[2]Leonid.Elshin@tatar.ru

***Abstract*- Supply chain management is considered as one of the key elements to leveraging a company's success. Active global crypto currency market growth "overwhelming" national economic systems contributes to the formation of a new type of economic relations which makes supply chain as the crucial factor in development. Despite the fact that at the current time in the world community a single (unified) approach to the legal regulation of the crypto currency market has not yet been developed, crypto currency is considered by many world regulators as a promising tool in the monetary policy of national economies. In this regard, it seems extremely urgent to solve the problem which reveals the features of the studied market development, as well as the possibility of its forecasting for short- and medium-term periods of time. The research subject is the process of economic and mathematical modelling of time series characterizing the bitcoin exchange rate volatility, based on the use of artificial neural networks. The purpose of the work is to search and scientifically substantiate the tools and mechanisms for developing prognostic estimates of the crypto currency market development. The paper considers the task of financial time series trend forecasting using the LSTM neural network for supply chain strategies. The time series composed of the BTC / USD currency pair data is analysed; the period analysed is a day. The authors analysed the neural network architecture, built a neural network model taking into account the heterogeneity and random volatility of the time series, developed and implemented an algorithm for solving the problem in the Python system. For training the neural network, data were used for the period from September 24, 2013, to March 17, 2019 (a total of 2002 data sets). The experiment boils down to that the constructed neural network model is trying to determine the trend of the time series for one next timeframe. The training was conducted with a "teacher". To determine the prediction error, the root-mean-square error (RMSE) was calculated. The results of the study are of practical interest for both government authorities in the field of crypto currency market regulation and for representatives of the business community, who are integrated or planning to integrate into the global crypto currency transaction system.**

Key words- *Machine learning, Supply chain management, time series; forecasting; artificial neural networks; the architecture of artificial neural networks; LSTM.*

1. Introduction

Deep learning (DL) is a subset of machine learning methods that use artificial neural networks (ANNs) built on the basis of an analogy with the structure of human brain neurons. A relatively small number of ingenious methods are used with great success in various fields (processing of images, text, video, speech, etc.), which made it possible to achieve significant progress compared with the results achieved over the previous decades. The deep learning owes its success to the availability of large volumes of training data and graphic processors (GPUs) which made it possible to construct a very efficient calculation procedure. At Google, Microsoft, Amazon, Apple, Facebook, and many others, deep learning methods are constantly used to analyse large amounts of data. Long- and short-term memory networks (LSTM) for short, can be used to predict time series. There are many types of LSTM models that can be used for each specific time series forecasting task. This paper is developed to LSTM models for predicting time series characterizing the dynamics of the Bitcoin exchange rate (BTC / USD).

1.1. Supply Chain Analytics

In the past few years, BDA has proven to be a true advantage for decision support systems, encouraging SC managers to employ these new techniques in their chain. The use of advanced BDA in a Supply Chain, also called Supply Chain Analytics [1- 5] (SCA), encompasses three main branches: Descriptive analytics, Predictive analytics, and Prescriptive analytics [6-8].

2. Methods

A perceptron is a simple algorithm that receives an input vector x containing n values $(x_1, x_2, \dots, x_n)$

which are often called input features, and the output gives two values: 1 (yes) or 0 (no). Formally speaking, the function is defined as follows:

$$f(x) = \begin{cases} 1, \text{если } wx + b > 0 \\ 0 \text{ если } wx + b \leq 0, \end{cases}$$

Where w is the weight vector, ws is the scalar product, and b is the displacement.

A Multilayer Perceptron (MSP) is a model with several linear layers. Figure 1 shows a neural network with one input, one intermediate and one output layer.

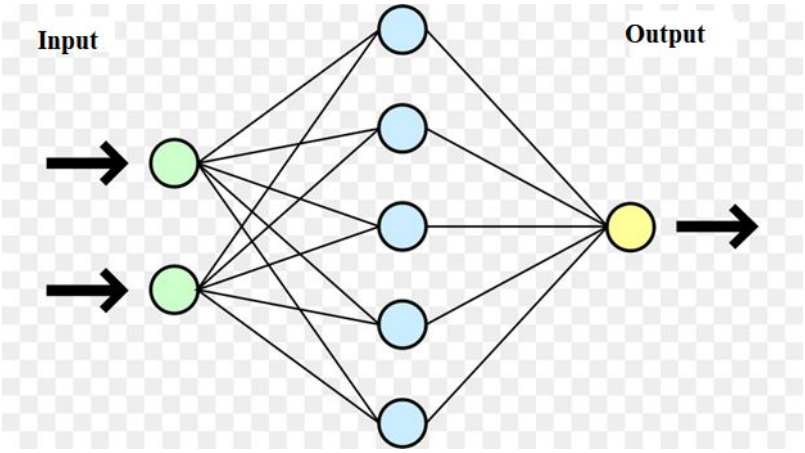

Figure1. Multilayer perceptron

Each node of the first layer receives an input signal and is excited in accordance with predefined local boundary conditions. The output of the first layer is fed to the input of the second and the output of the second to the last layer, consisting of one neuron. Recurrent neural networks are a class of neural networks that use input data of the consistent nature. Conventional neural networks (previously described multilayer perceptrons) have a fixed number of inputs and perceive each of them as an independent. In recurrent networks, communication between neurons can go not only from the lower layer to the upper but also from the neuron to itself, more precisely, to the previous value of this neuron itself or other neurons of the same layer. This allows us to reflect the dependence of the variable on its values at different points in time: the neuron learns to use not only the current input and what the neurons of the previous levels did with it, but also what happened to it by itself and, possibly, other neurons at the previous inputs. The input can be text, speech, time series, and generally any object in which the appearance of an element or sequence depends on previous elements. Recurrent neural networks can be considered as a graph consisting of unit cells, each of which performs the same operation for each element of the sequence. Recurrent neural networks are highly flexible and are used to solve problems such as speech recognition, language modelling, machine translation, and analysis of emotional colouring, image signing and many others. Recurrent neural networks can be adapted to various types of tasks by changing the configuration of cells in the graph. For time series, for example, quotes of currencies, dependence on past data is characteristic: this phenomenon is called a long-term trend. In cells of recurrent neural networks, this dependence is represented using a hidden state, or memory, in which a summary of past information is stored. The value of the hidden state at any time is a function of its value in the previous step and the data value in the current step:

$$h_t = \varphi(h_{t-1}, x_t),$$

Where h_t and h_{t-1} are latent state values at steps t and t - 1, respectively, and x_t is the input value at time t. Note that this equation is recurrent, i.e. h_{t-1} can be expressed through h_{t-2} and x_{t-1} etc., until we get to the start of the sequence. Thus, in recurrent neural networks, information on an arbitrarily long sequence is encoded and stored.

Figure 2 shows a cell of a recurrent neural network. At time t, the cell receives the value at the input x_t and displays the value y_t.. Part of y_t. (hidden state of h_t) is fed back to the input of the cell for use at the next step t + 1. If the parameters of a traditional neural network are stored in a weight matrix, then the parameters of a recurrent neural network are defined by three weight matrices U, V, and W corresponding to the input, output, and latent state.

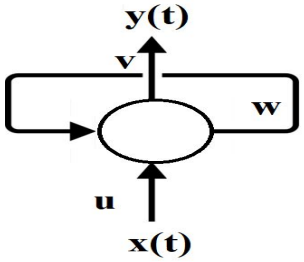

Figure 2. A cell of a recurrent neural network.

Figure 3 shows a recurrent neural network with three layers, suitable for processing a sequence with three elements. Note that the matrices U, V, W are divided between all steps since at each step the same operations are applied to different data. Thanks to the use of the same scales at all time steps, it is possible significantly to reduce the number of trained parameters of the recurrent neural network.

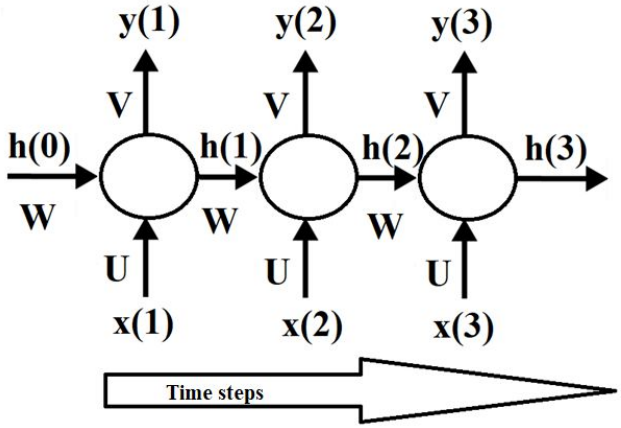

Figure3. Architecture of the recurrent neural network.

The internal state of a recurrent neural network at time t is determined by the value of the vector h_t equal to the result of applying the nonlinearity tanh to the sum of the product of the weight matrix W by the hidden state h_{t-1} at time t - 1 and the product of the weight matrix U by the input value x_t at time t. The choice of tanh nonlinearity, and not some other, is connected with the fact that its second derivative decreases very slowly, approaching to zero. Therefore, the gradients remain in the linear part of the activation function, which helps to cope with the disappearing gradient problem. Output vector y_t at time t is equal to the result of applying the softmax function to the product of the weight matrix V by the hidden state h_t and represents a set of exit probabilities:

$$\boldsymbol{h_t = \tanh\,(Wh_{t-1} + Ux_t),}$$

$$\boldsymbol{y_t = softmax(Vh_t).}$$

Since the same parameters are used in all steps, the gradient in each input depends not only on the current time series but also on the previous ones.

Learning a recurrent neural network involves an inverse distribution [1-4]. In the process of inverse distribution at each time step, the gradients of the loss function are calculated by the parameters U, V and W, and the sum of the gradients is used to update the parameters.

In ordinary recurrent neural networks, the problem of a vanishing and explosive gradient arises [5]. Because of this effect, it turns out that gradients at individual steps make no contribution to the learning process, so a regular recurrent neural network cannot take into account long-term dependencies.

Explosive gradients are detected more easily because when the gradient becomes too large and turns into a NaN, the learning process crashes. Gradient growth can be controlled by trimming them when a given threshold is reached [6].

There are several approaches to mitigate the problem of fading gradients, in particular, good initialization of W, but the most popular architecture is LSTM. It is specifically designed to deal with a fading gradient and is more effectively trained in long-term dependencies.

LSTM is a variant of a recurrent neural network that can learn long-term dependencies. [7] LSTM networks work well for a wide range of tasks and are the most popular type of recurrent neural network. In LSTM, a combination of latency in the previous step and the current input data in a layer with activation function are used for the implementation of recurrence. Figure 4 shows the transformations applied to the latent state on the timeline t.

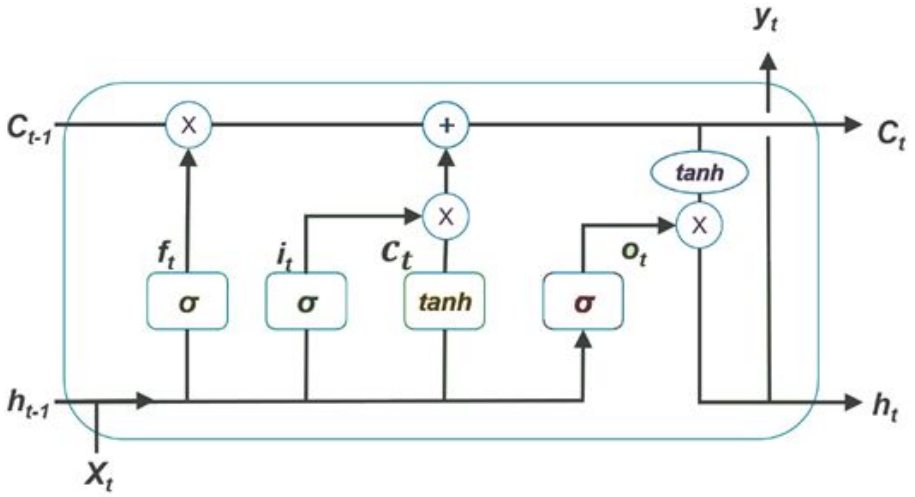

Figure4. LSTM network architecture

The horizontal line at the top shows the state of cell c; it represents the internal memory of the block. The latent state is shown on the line below, and the gates f, i, c, o are the mechanisms due to which the LSTM network bypasses the problem of the vanishing gradient. In the learning process, LSTM finds the parameters of the following valves:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f),$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i),$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o),$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c),$$
$$h_t = o_t \circ \sigma_c(c_t),$$

x_t - input vector

y_t - output vector

c_t - state vector

f_t - vector of forgetting gate, the weight of remembering old information,

i_t - vector of the input gate, the weight of obtaining new information,

o_t - gate of the output vector,

σ_g - an activation function based on a signoid,

σ_c - activation function based on a hyperbolic tangent.

3. Results and discussion

To build a neural network, the predictive crypto currency rate (BTC / USD) for one day in advance was taken from [6].

Input data:

Low - the lowest price in a period of time t,

High - the highest price in a period of time t,

Open - the opening price in a period of time t,

Close - closing price at time t,

Volumefrom - trading volume in bitcoins in a period of time t,

Volumeto - trading volume in dollars over a period of time t.

Output

Low - the lowest price in the period t + 1,

High - the highest price in the period t + 1,

Close - closing price in the period t + 1.

To improve the operation of neural networks, the authors used minimax data normalization within [0: 1] [8].

The data obtained were divided into training and test (1600 training and 401 test). The training set is used to find the relationship between input and output variables, while test data evaluate model performance. The assignment of data for training and for tests is performed using random sampling.

A neural network with one hidden LSTM layer was built.

An important task in building a neural network is the correct selection of the loss function, optimizer and parameters. To select the parameters, the random search method [9] [10] was used. To estimate the reliability of the constructed model, the root mean square error (RMSE) was calculated. Table 1 presents the fifteen best experimental results.

Table1. The results of the 15 best-conducted experiments evaluating the effectiveness of the constructed neural network with one hidden LSTM layer

loss	activation	optimizer	neurons	epochs	batch_siz	dropout	rmse	time
logcosh	relu	Nadam	16	488	216	17.00	139.35	24,17871
logcosh	linear	Nadam	26	310	100	19.00	139.87	28,24852
mse	linear	Nadam	6	381	80	3.00	141.21	37.89283
mse	linear	Nadam	14	373	80	8.00	141.34	31,2856
mae	tanh	Adadelta	5	494	110	11.00	141.71	30,01612
logcosh	tanh	Nadam	30	382	144	0.00	142.44	28,1057
mae	linear	Nadam	12	455	123	4.00	142.48	25.54287
mse	selu	Nadam	21	231	116	1.00	142.95	18.79688

logcosh	selu	Nadam	8	486	115	2.00	143.25	35.19759
mse	elu	Nadam	22	449	144	15.00	143.39	24,16052
logcosh	elu	Rmsprop	10	429	137	17.00	143.57	37,40822
msle	linear	Nadam	26	461	116	17.00	143.67	30,50476
mse	tanh	Rmsprop	17	480	106	9.00	143.92	30,08084
msle	tanh	Rmsprop	24	460	71	14.00	143.97	45.33771
mae	relu	Nadam	16	369	192	16.00	144.20	16,20398

4. Conclusion

Supply chain process is need to extract relevant information that will lead to a competitive advantage. The following conclusions can be drawn from the experiments:

- the constructed recurrent neural network is well suited for predicting the price of the BTC / USD currency pair (this follows from the “rmse” column of Table 1);
- error levels in the training and validation samples show that the model predicts real data well.

The main disadvantages of simple recurrent networks considered are the problem of disappearing and explosive gradients. The architecture of a simple recurrent network and the LSTM network were considered, a separate cell of a simple recurrent network was considered. The authors considered the task of predicting the exchange rate for the BTC / USD currency pair with a daily time frame. A selection was made using random search.

Acknowledgements

The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University. The publication was prepared as part of the scientific project supported by the Russian Foundation for Basic Research No. 18-010-00536 A.

References

[1] E. V. Arkhangelskaya, A. Kadurin, S. I. Nikolenko “*Deep Learning. Immersion in the world of neural networks*”, Publishing house “Piter”, 2018.

[2] V.V. Strizhov “*Error function in regression restoration problems*”. Factory Laboratory, p. 5, 2013.

[3] G.E. Hinton, D.E. Rumelhart, R.J. Williams, *Learning Internal Representations by Backpropagating error*, Parallel Distributes Processing: Explorations in the Microstructure of Cognition, 1985.

[4] V. S. Tormozov “*Investigation of Neural Networks of Memory*”, Pacific State University, Khabarovsk, 2016.

[5] J.L. ELMAN “*Finding Structure in Time*” University of California, San Diego, 1990.

[6] R. Pascanumu, T. Mikolov, Y. Bengio "*On the Difficulty of Training Recurrent Neural Networks*", 2013.

[7] S. Hochreiter and J. Schmidhuber, "*Long short-term memory*", Neural Computation, 1997.

[8] V.V. Golenkov, O. M. Gerget "*Analysis of methods for normalizing heterogeneous data for analysis in expert systems of medical diagnostics*. "Tomsk Polytechnic University, Department of Applied Mathematics, 2013.

[9] J. Bergstra, Y. Bengio, *“Random Search for Hyper-Parameter Optimization*,” Departement d'Informatique et de recherche op´erationnelle ’Universite de Montr´eal ´ Montreal, QC, H3C 3J7, Canada, 2012.

[10] V. T. Kusherbayeva, Yu. A. Sushkov “*Statistical Investigation of the Random Search Algorithm*”, St. Petersburg State University, p. 5, 2007.