Forecast Method Based on the Time-Delay Mean Field Boltzmann Machine

Oleg Grygor, Eugene Fedorov and Olga Nechyporenko

Cherkasy State Technological University, Shevchenko blvd., 460, Cherkasy, 18006, Ukraine

Abstract

The problem of insufficient forecast efficiency for supply chain management is solved. A neural network forecast model based on the Time-Delay Mean Field Boltzmann Machine with time delays in the visible layer has been created. In the process of adjusting the structure of the developed model, the length of the hidden layer was determined, and the calculation of the model parameters was carried out on the basis of the parallel computing platform CUDA. Improving forecast accuracy and speed of calculations makes it possible to improve the quality of the forecast, resulting in increased supply flexibility and reduced logistics costs. A software toolkit based on the Matlab package has been developed, which makes it possible to implement the proposed method. The developed software tools are used to solve the problem of supply chains forecasting.

Keywords

Forecast efficiency, supply chain management problem, neural network forecast model, Time-Delay Mean Field Boltzmann Machine, positive and negative learning phase

1. Introduction

Supply chains are complex adaptive systems characterized by structural and dynamic complexity, operating under a large number of random factors. Supply chain management is based on forecasting the demand for the final product. This requires efficient and intelligent supply chain planning. Planning challenges include, but are not limited to, fragmented data across the organization and difficulty in forecasting deliveries. This leads to low accuracy of sales plans, a large volume of illiquid products and, as a result, to losses for the company.

The tasks of reducing inventory and increasing turnover are directly related to the accuracy of forecasting sales. When calculating safety stock, the average deviation of sales from forecasts is one of the main components. Today, one of the main problems in the field of supply chain management is the lack of forecast efficiency [1-4]. Therefore, the decisions made may not be accurate and fast enough. Improving forecast accuracy can lead to an increase in inventory turnover, as well as increase sales due to a decrease in the number of out-of-stock. Thus, the creation of effective forecasting methods for supply chain management is an urgent task.

There is a set of methods as a means for forecasting, among which are:

- logical forecasting methods based on classification and regression trees [5];
- forecasting methods based on exponential smoothing [6];
- regression and autoregressive forecasting methods [7];
- neural network forecasting methods [8, 9];
- structural forecasting methods based on Markov chains [10].

Using artificial neural networks for forecasting provides the following advantages:

- assumptions about the distribution of input features are not required;
- analysis of systems with a high degree of nonlinearity is possible;

Information Technology and Implementation (IT&I-2021), December 01-03, 2021, Kyiv, Ukraine
EMAIL: chdtu-cherkassy@ukr.net (O. Grygor); fedorovee75@ukr.net (E. Fedorov); olne@ukr.net (O. Nechyporenko)
ORCID: 0000-0002-5233-290X (O. Grygor); 0000-0003-3841-7373 (E. Fedorov); 0000-0002-3954-3796 (O. Nechyporenko)

- high adaptability;
- rapid model development;
- the relationships between the input features are investigated on ready-made models;
- a priori information about the input features may be missing;
- the original data may be incomplete or contain noise, as well as highly correlated;
- analysis of systems with a large number of input features is possible;
- analysis of systems with heterogeneous characteristics is possible;
- a complete enumeration of all possible models is not required.

Therefore, a neural network forecasting method will be used in the article.

2. Formal problem statement

Let the training set $S=\{(x_\mu, d_\mu)\}$, $\mu \in \overline{1,P}$ be given for the forecast.

Then the problem of improving the forecast accuracy for the Time-Delay Mean Field Boltzmann Machine (TDMFBM) model is $g(x,W)$, where x – is the input vector, W – is the vector of parameters, represented as the problem of finding such a vector of parameters W^* for this model, that satisfies criterion $F=\frac{1}{P}\sum_{\mu=1}^{P}(g(x_\mu,W^*)-d_\mu)^2 \to \min$.

The aim of the work is to create an effective forecasting method for supply chain management. To achieve this goal, the following tasks were set and solved:

- analyze existing neural network forecasting methods;
- create a neural network forecast model based on the mean field Boltzmann machine;
- choose a criterion for evaluating the effectiveness of a neural network forecast model based on the mean field Boltzmann machine;
- develop a method for identifying the parameters values of the neural network forecast model based on the mean field Boltzmann machine;
- perform numerical studies.

3. Literature review

The number of publications demonstrates significant attention to the application of advanced analytics, methods and modern computer tools of artificial intelligence in the field of supply chain management, but also leaves unresolved and insufficiently studied a number of problems regarding the development and synthesis of methods and models of artificial intelligence.

The most commonly used forecast neural networks are:

1. Gateway neural networks:
- long short-term memory (LSTM) [11, 12];
- bidirectional long short-term memory (BLSTM) [13, 14];
- gateway recurrent unit (GRU) [15-17];
- bidirectional gateway recurrent unit (BGRU) [18, 19].

2. Reservoir neural networks:
- echo state network (ESN) [20, 21];
- liquid state machine (LSM) [22-24].

Table 1 shows the comparative characteristics of forecasting neural networks.

The learning rate is directly proportional to the computational complexity. For LSTM computational complexity ~$PN^{(1)}(5M^{(0)}+3M^{(0)}S+24S+S^2)$, for BLSTM computational complexity ~$2PN^{(1)}(5M^{(0)}+3M^{(0)}S+24S+S^2))$, for GRU computational complexity ~$PN^{(1)}6(M^{(0)}+N^{(1)})$, for BGRU computational complexity ~$PN^{(1)}6(M^{(0)}+N^{(1)})$, for ESN computational complexity ~$PN^{(1)}(M^{(0)}+N^{(1)})+(\max\{P,M^{(0)}+N^{(1)}\})^2$, for LSM computational complexity ~$PN^{(r)}(N^{(r)}M^{(0)}+N^{(1)})$, where $M^{(0)}$ – the number of unit delays for the input layer, S – the number of cell, $N^{(1)}$ – the number of neurons in the first layer, $N^{(r)}$ – the number of neurons in the reservoir layer, P – training set cardinality, $N^{(1)}<<P$, $N^{(r)}<<P$. According to Table 1, none of the networks meets all the criteria.

Thereby, the creation of a neural network that will eliminate the specified drawback is relevant.

Table 1

Comparative characteristics of forecasting neural networks

Criterion \ Network	LSTM	BLSTM	GRU	BGRU	ESN	LSM
The presence of feedback	+	+	+	+	+	+
Low probability of getting into a local extremum	-	-	-	-	-	-
High learning speed	-	-	-	-	-	-
Possibility of batch training	-	-	-	-	-	-

4. Materials and methods

4.1. Block diagram of a neural network forecast model

Figures 1-2 show a block diagram of a forecast model based on the Time-Delay Mean Field Boltzmann Machine (TDMFBM) of two types, which is a recurrent neural network with one hidden layer.

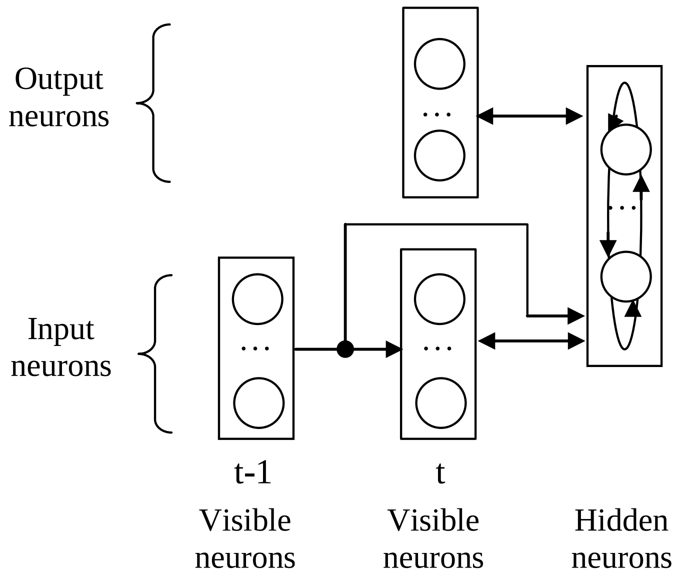

Figure 1: Block diagram of the forecast model based on the Time-Delay Mean Field Boltzmann Machine with time delays for the input neurons of the visible layer (TDMFBM type 1)

In contrast to the traditional mean field Boltzmann machine (MFBM) [25, 26], time delays are used for the neurons of the visible layer, and the neurons of the visible layer are not connected with each other. TDMFBM type 1 has time delays in the input layer. TDMFBM type 2 has time delays in the input and output layers.

4.2. Forecasting model based on TDMFBM type 1

***Positive phase* (steps 1-3)**

1. Initial time step

$$\mu = 1.$$

Initialization of the state of the visible input neurons of the time delay

$$\mathbf{x}^{in}(\mu - t) = 0,\ t \in \overline{1, M^{in}}.$$

2. Initialization of the state of visible input, hidden and output neurons

$$\mathbf{x}^{in}(\mu) = \mathbf{x}^{in}_{\mu},\ \mathbf{x}^{h}(\mu) = \mathbf{0},\ \mathbf{x}^{out}(\mu) = \mathbf{0}.$$

3. Computation of the state of hidden neurons ($j \in \overline{1, N^h}$) at time μ

$$s^h_j(\mu) = b^h_j + \sum_{t=0}^{M^{in}} \sum_{i=1}^{N^{in}} w^{in-h}_{tij} x^{in}_i(\mu - t) + \sum_{i=1}^{N^{out}} w^{out-h}_{ij} x^{out}_i(\mu) + \sum_{i=1}^{N^h} w^{h-h}_{ij} x^h_i(\mu),$$

$$x^h_j(\mu) = \frac{1}{1 + \exp\left(-s^h_j(\mu)\right)},$$

where w^{in-h}_{tij} – synaptic weights between the visible input layer (taking into account unit delays) and the hidden layer,

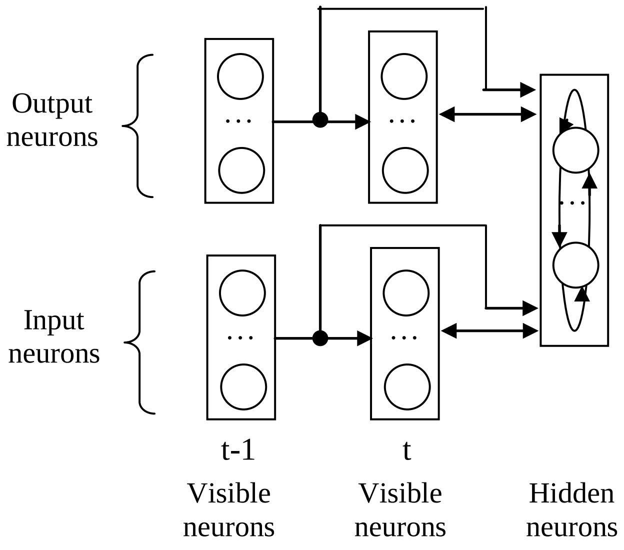

Figure 2: Block diagram of the forecast model based on the Time-Delay Mean Field Boltzmann Machine with time delays for the input and output neurons of the visible layer (TDMFBM type 2)

w^{out-h}_{ij} – synaptic weights between the visible output and the hidden layer,

w^{h-h}_{ij} – synaptic weights inside the hidden layers,

b^h_j – bias of neurons of the hidden layer,

M^{in} – the number of unit delays for the visible input layer,

N^{in} – the number of neurons in the input layer,

N^h – the number of neurons in the hidden layer,

N^{out} – the number of neurons in the output layer.

***Negative phase* (step 4)**

4. Computation of the state of visible output neurons ($j \in \overline{1, N^{out}}$) at time μ

$$s^{out}_j(\mu) = b^{out}_j + \sum_{t=0}^{M^{in}} \sum_{i=1}^{N^{in}} w^{in-out}_{tij} x^{in}_i(\mu - t) + \sum_{i=1}^{N^h} w^{out-h}_{ij} x^h_i(\mu),$$

$$x^{out}_j(\mu) = \frac{1}{1 + \exp\left(-s^{out}_j(\mu)\right)},$$

where b^{out}_j – bias of neurons of the visible output layer.

The result is vector $(x^{out}_1(\mu), ..., x^{out}_{N^{out}}(\mu))$.

4.3. Forecasting model based on TDMFBM type 2

Positive phase **(steps 1-3)**

1. Initial time step

$$\mu = 1.$$

Initialization of the state of the visible input and output neurons of the time delay

$$\mathbf{x}^{in}(\mu - t) = 0,\ t \in \overline{1, M^{in}},\quad \mathbf{x}^{out}(\mu - t) = 0,\ t \in \overline{1, M^{out}}.$$

2. Initialization of the state of visible input and output neurons

$$\mathbf{x}^{in}(\mu) = \mathbf{x}^{in}_{\mu},\ \mathbf{x}^{h}(\mu) = \mathbf{0},\ \mathbf{x}^{out}(\mu) = \mathbf{0}.$$

3. Computation of the state of hidden neurons ($j \in \overline{1, N^h}$) at time μ

$$s^h_j(\mu) = b^h_j + \sum_{t=0}^{M^{in}} \sum_{i=1}^{N^{in}} w^{in-h}_{tij} x^{in}_i(\mu - t) + \sum_{t=0}^{M^{out}} \sum_{i=1}^{N^{out}} w^{out-h}_{tij} x^{out}_i(\mu - t) + \sum_{i=1}^{N^h} w^{h-h}_{ij} x^h_i(\mu),$$

$$x^h_j(\mu) = \frac{1}{1 + \exp\left(-s^h_j(\mu)\right)},$$

where w^{in-h}_{tij} – synaptic weights between the visible input layer (taking into account unit delays) and the hidden layer,

w^{out-h}_{tij} – synaptic weights between the visible output layer (taking into account unit delays) and the hidden layer,

w^{h-h}_{ij} – synaptic weights inside the hidden layers,

b^h_j – bias of neurons of the hidden layer,

M^{in} – the number of unit delays for the visible input layer,

M^{out} – the number of unit delays for the visible output layer,

N^{in} – the number of neurons in the input layer,

N^h – the number of neurons in the hidden layer,

N^{out} – the number of neurons in the output layer.

Negative phase **(step 4)**

4. Computation of the state of visible output neurons ($j \in \overline{1, N^{out}}$) at time μ

$$s^{out}_j(\mu) = b^{out}_j + \sum_{t=0}^{M^{in}} \sum_{i=1}^{N^{in}} w^{in-out}_{tij} x^{in}_i(\mu - t) + \sum_{i=1}^{N^h} w^{out-h}_{0ij} x^h_i(\mu),$$

$$x^{out}_j(\mu) = \frac{1}{1 + \exp\left(-s^{out}_j(\mu)\right)},$$

where b^{out}_j – bias of neurons of the visible output layer.

The result is vector $(x^{out}_1(\mu), \ldots, x^{out}_{N^{out}}(\mu))$.

4.4. Criterion for evaluating the effectiveness of a neural network forecast model

In this work, for training the TDMFBM model, a model adequacy criterion was chosen, which means the choice of such values of parameters $W = \{w^{in-h}_{tij},\ w^{out-h}_{tij},\ w^{in-in}_{tij},\ w^{out-out}_{tij}\}$, which deliver a minimum of the mean squared error (the difference between the model output and the desired output):

$$F = \frac{1}{P}\sum_{\mu=1}^{P}\sum_{i=1}^{N^{out}}(x_{\mu i}^{out} - d_{\mu i})^2 \to \min_{W} . \qquad (1)$$

The training of the TDMFBM model is subject to criterion (1).

4.5. Method for determining the parameter values of the forecasting model based on TDMFBM type 1

1. Number of training iteration $n=1$, initialization by means of uniform distribution on the interval (0,1) or [-0.5, 0.5] bias $b_i^{out}(n)$, $i \in \overline{1, N^{out}}$, $b_j^h(n)$, $j \in \overline{1, N^h}$, and weights $w_{tij}^{in-h}(n)$, $t \in \overline{0, M^{in}}$, $i \in \overline{1, N^{in}}$, $j \in \overline{1, N^h}$, $w_{tij}^{in-out}(n)$, $t \in \overline{0, M^{in}}$, $i \in \overline{1, N^{in}}$, $j \in \overline{1, N^{out}}$, $w_{ij}^{out-h}(n)$, $i \in \overline{1, N^{out}}$, $j \in \overline{1, N^h}$, $w_{ij}^{h-h}(n)$, $i \in \overline{1, N^h}$, $j \in \overline{1, N^h}$, $w_{tii}^{in-h}(n)=0$, $w_{tij}^{in-out}(n)=0$, $w_{ii}^{out-h}(n)=0$, $w_{ii}^{h-h}(n)=0$, $w_{tij}^{in-h}(n)=w_{tji}^{in-h}(n)$, $w_{tij}^{in-out}(n)=w_{tji}^{in-out}(n)$, $w_{ij}^{out-h}(n)=w_{ji}^{out-h}(n)$, $w_{ij}^{h-h}(n)=w_{ji}^{h-h}(n)$, where M^{in} is the number of unit delays for visible input neurons.

2. A training set $\{(\mathbf{x}_\mu^{in}, \mathbf{x}_\mu^{out}) \mid \mathbf{x}_\mu^{in} \in (0,1)^{N^{in}}, \mathbf{x}_\mu^{out} \in (0,1)^{N^{out}}\}$, $\mu \in \overline{1, P}$ is set, where $\mathbf{x}_\mu^{in}$ – μ th training vector of states of visible input neurons, $\mathbf{x}_\mu^{out}$ – μ th training vector of states of visible output neurons, P – is the power of the training set.

Positive phase **(steps 3-6)**

3. Initial time step

$$\mu = 1.$$

Initialization of the state of the visible input neurons of the time delay

$$\mathbf{x}^{in}(\mu - t) = 0, \ \mathbf{x1}^{in}(\mu - t) = 0, \ t \in \overline{1, M^{in}} .$$

4. Initialization of the state of visible input and output neurons

$$\mathbf{x}^{in}(\mu) = \mathbf{x}_\mu^{in}, \ \mathbf{x}^h(\mu) = \mathbf{0}, \ \mathbf{x}^{out}(\mu) = \mathbf{x}_\mu^{out} .$$

5. Computation of the state of hidden neurons ($j \in \overline{1, N^h}$) at time μ

$$s_j^h(\mu) = b_j^h(n) + \sum_{t=0}^{M^{in}}\sum_{i=1}^{N^{in}} w_{tij}^{in-h}(n) x_i^{in}(\mu - t) + \sum_{i=1}^{N^{out}} w_{ij}^{out-h}(n) x_i^{out}(\mu) + \sum_{i=1}^{N^h} w_{ij}^{h-h}(n) x_i^h(\mu),$$

$$x_j^h(\mu) = \frac{1}{1 + \exp\left(-s_j^h(\mu)\right)} .$$

6. Preservation of the state of neurons in the positive phase at time μ, i.e. $\mathbf{x1}^{in}(\mu) = \mathbf{x}^{in}(\mu)$, $\mathbf{x1}^{out}(\mu) = \mathbf{x}^{out}(\mu)$, $\mathbf{x1}^h(\mu) = \mathbf{x}^h(\mu)$. If $\mu < P$, then $\mu = \mu + 1$, go to 4.

Negative phase **(steps 7-11)**

7. Initial time step

$$\mu = 1.$$

Initialization of the state of the visible input neurons of the time delay

$$\mathbf{x}^{in}(\mu - t) = 0, \ \mathbf{x2}^{in}(\mu - t) = 0, \ t \in \overline{1, M^{in}} .$$

8. Initialization of the state of visible input and output, hidden neurons

$$\mathbf{x}^{in}(\mu) = \mathbf{x1}^{in}(\mu), \ \mathbf{x}^{out}(\mu) = \mathbf{x1}^{out}(\mu), \ \mathbf{x}^h(\mu) = \mathbf{x1}^h(\mu) .$$

9. Computation of the state of visible output neurons ($j \in \overline{1, N^{out}}$) at time μ

$$s_j^{out}(\mu)=b_j^{out}(n)+\sum_{t=0}^{M^{in}}\sum_{i=1}^{N^{in}}w_{tij}^{in-out}(n)x_i^{in}(\mu-t)+\sum_{i=1}^{N^h}w_{ij}^{out-h}(n)x_i^h(\mu),$$

$$x_j^{out}(\mu)=\frac{1}{1+\exp\left(-s_j^{out}(\mu)\right)}.$$

10. Computation of the state of hidden neurons ($j\in\overline{1,N^h}$) at time μ

$$s_j^h(\mu)=b_j^h(n)+\sum_{t=0}^{M^{in}}\sum_{i=1}^{N^{in}}w_{tij}^{in-h}(n)x_i^{in}(\mu-t)+\sum_{i=1}^{N^{out}}w_{ij}^{out-h}(n)x_i^{out}(\mu)+\sum_{i=1}^{N^h}w_{ij}^{h-h}(n)x_i^h(\mu),$$

$$x_j^h(\mu)=\frac{1}{1+\exp\left(-s_j^h(\mu)\right)}.$$

11. Saving the state of neurons in the negative phase at time μ, i.e. $\mathbf{x}2^{in}(\mu)=\mathbf{x}^{in}(\mu)$, $\mathbf{x}2^{out}(\mu)=\mathbf{x}^{out}(\mu)$, $\mathbf{x}2^h(\mu)=\mathbf{x}^h(\mu)$. If $\mu<P$, then $\mu=\mu+1$, go to 8.

12. Adjustment of synaptic weights and bias based on Boltzmann's rule

$$b_i^{out}(n)=b_i^{out}(n)+\eta\left(\frac{1}{P}\sum_{\mu=1}^{P}x1_i^{out}(\mu)-\frac{1}{P}\sum_{\mu=1}^{P}x2_i^{out}(\mu)\right),\ i\in\overline{1,N^{out}},$$

$$b_j^h(n)=b_j^h(n)+\eta\left(\frac{1}{P}\sum_{\mu=1}^{P}x1_j^h(\mu)-\frac{1}{P}\sum_{\mu=1}^{P}x2_j^h(\mu)\right),\ j\in\overline{1,N^h},$$

$$w_{tij}^{in-h}(n)=w_{tij}^{in-h}(n)+\eta(\rho_{tij}^{+}-\rho_{tij}^{-}),\ t\in\overline{0,M^{in}},\ i\in\overline{1,N^{in}},\ j\in\overline{1,N^h},$$

$$\rho_{tij}^{+}=\frac{1}{P}\sum_{\mu=1}^{P}x1_i^{in}(\mu-t)x1_j^h(\mu),\quad \rho_{tij}^{-}=\frac{1}{P}\sum_{\mu=1}^{P}x2_i^{in}(\mu-t)x2_j^h(\mu),$$

$$w_{tij}^{in-out}(n)=w_{tij}^{in-out}(n)+\eta(\rho_{tij}^{+}-\rho_{tij}^{-}),\ t\in\overline{0,M^{in}},\ i\in\overline{1,N^{in}},\ j\in\overline{1,N^{out}},$$

$$\rho_{tij}^{+}=\frac{1}{P}\sum_{\mu=1}^{P}x1_i^{in}(\mu-t)x1_j^{out}(\mu),\quad \rho_{tij}^{-}=\frac{1}{P}\sum_{\mu=1}^{P}x2_i^{in}(\mu-t)x2_j^{out}(\mu),$$

$$w_{ij}^{out-h}(n)=w_{ij}^{out-h}(n)+\eta(\rho_{ij}^{+}-\rho_{ij}^{-}),\ i\in\overline{1,N^{out}},\ j\in\overline{1,N^h},$$

$$\rho_{ij}^{+}=\frac{1}{P}\sum_{\mu=1}^{P}x1_i^{out}(\mu)x1_j^h(\mu),\quad \rho_{ij}^{-}=\frac{1}{P}\sum_{\mu=1}^{P}x2_i^{out}(\mu)x2_j^h(\mu),$$

$$w_{ij}^{h-h}(n)=w_{ij}^{h-h}(n)+\eta(\rho_{ij}^{+}-\rho_{ij}^{-}),\ i,j\in\overline{1,N^h},$$

$$\rho_{ij}^{+}=\frac{1}{P}\sum_{\mu=1}^{P}x1_i^h(\mu-t)x1_j^h(\mu),\quad \rho_{ij}^{-}=\frac{1}{P}\sum_{\mu=1}^{P}x2_i^h(\mu-t)x2_j^h(\mu).$$

13. If $\frac{1}{P}\sum_{\mu=1}^{P}\left(\sum_{i=1}^{N^{out}}|\,x1_i^{out}(\mu)-x2_i^{out}(\mu)\,|\right)>\varepsilon$, then $n=n+1$, go to 2.

4.6. Method for determining the parameter values of the forecasting model based on TDMFBM type 2

1. Number of training iteration $n=1$, initialization by means of uniform distribution on the interval (0,1) or [-0.5, 0.5] bias $b_i^{out}(n)$, $i\in\overline{1,N^{out}}$, $b_j^h(n)$, $j\in\overline{1,N^h}$, and weights $w_{tij}^{in-h}(n)$, $t\in\overline{0,M^{in}}$, $i\in\overline{1,N^{in}}$, $j\in\overline{1,N^h}$, $w_{tij}^{in-out}(n)$, $t\in\overline{0,M^{in}}$, $i\in\overline{1,N^{in}}$, $j\in\overline{1,N^{out}}$, $w_{tij}^{out-h}(n)$, $t\in\overline{0,M^{out}}$, $i\in\overline{1,N^{out}}$, $j\in\overline{1,N^h}$, $w_{ij}^{h-h}(n)$, $i\in\overline{1,N^h}$, $j\in\overline{1,N^h}$, $w_{tii}^{in-h}(n)=0$, $w_{tij}^{in-out}(n)=0$, $w_{tii}^{out-h}(n)=0$,

$w_{ii}^{h-h}(n)=0$, $w_{tij}^{in-h}(n)=w_{tji}^{in-h}(n)$, $w_{tij}^{in-out}(n)=w_{tji}^{in-out}(n)$, $w_{tij}^{out-h}(n)=w_{tji}^{out-h}(n)$, $w_{ij}^{h-h}(n)=w_{ji}^{h-h}(n)$, where M^{in} is the number of unit delays for visible input neurons, M^{out} is the number of single delays for visible output neurons.

2. A training set $\{(\mathbf{x}_\mu^{in},\mathbf{x}_\mu^{out}) \mid \mathbf{x}_\mu^{in}\in(0,1)^{N^{in}}, \mathbf{x}_\mu^{out}\in(0,1)^{N^{out}}\}$ is set, $\mu\in\overline{1,P}$, where $\mathbf{x}_\mu^{in}$ – μ th training vector of states of visible input neurons, $\mathbf{x}_\mu^{out}$ – μ th training vector of states of visible output neurons, P is the power of the training set.

Positive phase **(steps 3-6)**

3. Initial time step

$$\mu=1.$$

Initialization of the state of the visible input and output neurons of the time delay

$$\mathbf{x}^{in}(\mu-t)=0,\ \mathbf{x1}^{in}(\mu-t)=0,\ t\in\overline{1,M^{in}},\quad \mathbf{x}^{out}(\mu-t)=0,\ \mathbf{x1}^{out}(\mu-t)=0,\ t\in\overline{1,M^{out}}.$$

4. Initialization of the state of visible input and output neurons

$$\mathbf{x}^{in}(\mu)=\mathbf{x}_\mu^{in},\ \mathbf{x}^{h}(\mu)=\mathbf{0},\ \mathbf{x}^{out}(\mu)=\mathbf{x}_\mu^{out}.$$

5. Computation of the state of hidden neurons ($j\in\overline{1,N^h}$) at time μ

$$s_j^h(\mu)=b_j^h(n)+\sum_{t=0}^{M^{in}}\sum_{i=1}^{N^{in}}w_{tij}^{in-h}(n)x_i^{in}(\mu-t)+\sum_{t=0}^{M^{out}}\sum_{i=1}^{N^{out}}w_{tij}^{out-h}(n)x_i^{out}(\mu-t)+\sum_{i=1}^{N^h}w_{ij}^{h-h}(n)x_i^h(\mu),$$

$$x_j^h(\mu)=\frac{1}{1+\exp\left(-s_j^h(\mu)\right)}$$

6. Saving the state of neurons in a positive phase at time μ, i.e. $\mathbf{x1}^{in}(\mu)=\mathbf{x}^{in}(\mu)$, $\mathbf{x1}^{out}(\mu)=\mathbf{x}^{out}(\mu)$, $\mathbf{x1}^{h}(\mu)=\mathbf{x}^{h}(\mu)$. If $\mu<P$, then $\mu=\mu+1$, go to 4.

Negative phase **(steps 7-11)**

7. Initial time step

$$\mu=1.$$

Initialization of the state of the visible input and output neurons of the time delay

$$\mathbf{x}^{in}(\mu-t)=0,\ \mathbf{x2}^{in}(\mu-t)=0,\ t\in\overline{1,M^{in}},$$

$$\mathbf{x}^{out}(\mu-t)=0,\ \mathbf{x2}^{out}(\mu-t)=0,\ t\in\overline{1,M^{out}}.$$

8. Initialization of the state of visible input and output, hidden neurons

$$\mathbf{x}^{in}(\mu)=\mathbf{x1}^{in}(\mu),\ \mathbf{x}^{out}(\mu)=\mathbf{x1}^{out}(\mu),\ \mathbf{x}^{h}(\mu)=\mathbf{x1}^{h}(\mu).$$

9. Computation of the state of visible output neurons ($j\in\overline{1,N^{out}}$) at time μ

$$s_j^{out}(\mu)=b_j^{out}(n)+\sum_{t=0}^{M^{in}}\sum_{i=1}^{N^{in}}w_{tij}^{in-out}(n)x_i^{in}(\mu-t)+\sum_{i=1}^{N^h}w_{0ij}^{out-h}(n)x_i^h(\mu),$$

$$x_j^{out}(\mu)=\frac{1}{1+\exp\left(-s_j^{out}(\mu)\right)}.$$

10. Computation of the state of hidden neurons ($j\in\overline{1,N^h}$) at time μ

$$s_j^h(\mu)=b_j^h(n)+\sum_{t=0}^{M^{in}}\sum_{i=1}^{N^{in}}w_{tij}^{in-h}(n)x_i^{in}(\mu-t)+\sum_{t=0}^{M^{out}}\sum_{i=1}^{N^{out}}w_{tij}^{out-h}(n)x_i^{out}(\mu-t)+\sum_{i=1}^{N^h}w_{ij}^{h-h}(n)x_i^h(\mu),$$

$$x_j^h(\mu)=\frac{1}{1+\exp\left(-s_j^h(\mu)\right)}.$$

11. Saving the state of neurons in the negative phase at time μ, i.e. $\mathbf{x2}^{in}(\mu)=\mathbf{x}^{in}(\mu)$, $\mathbf{x2}^{out}(\mu)=\mathbf{x}^{out}(\mu)$, $\mathbf{x2}^{h}(\mu)=\mathbf{x}^{h}(\mu)$. If $\mu<P$, then $\mu=\mu+1$, go to 8.

12. Adjustment of synaptic weights and bias based on Boltzmann's rule

$$b_i^{out}(n) = b_i^{out}(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_i^{out}(\mu) - \frac{1}{P}\sum_{\mu=1}^{P} x2_i^{out}(\mu)\right),\ i \in \overline{1, N^{out}},$$

$$b_j^h(n) = b_j^h(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_j^h(\mu) - \frac{1}{P}\sum_{\mu=1}^{P} x2_j^h(\mu)\right),\ j \in \overline{1, N^h},$$

$$w_{tij}^{in-h}(n) = w_{tij}^{in-h}(n) + \eta(\rho_{tij}^{+} - \rho_{tij}^{-}),\ t \in \overline{0, M^{in}},\ i \in \overline{1, N^{in}},\ j \in \overline{1, N^h},$$

$$\rho_{tij}^{+} = \frac{1}{P}\sum_{\mu=1}^{P} x1_i^{in}(\mu - t)x1_j^h(\mu),\quad \rho_{tij}^{-} = \frac{1}{P}\sum_{\mu=1}^{P} x2_i^{in}(\mu - t)x2_j^h(\mu),$$

$$w_{tij}^{in-out}(n) = w_{tij}^{in-out}(n) + \eta(\rho_{tij}^{+} - \rho_{tij}^{-}),\ t \in \overline{0, M^{in}},\ i \in \overline{1, N^{in}},\ j \in \overline{1, N^{out}},$$

$$\rho_{tij}^{+} = \frac{1}{P}\sum_{\mu=1}^{P} x1_i^{in}(\mu - t)x1_j^{out}(\mu),\quad \rho_{tij}^{-} = \frac{1}{P}\sum_{\mu=1}^{P} x2_i^{in}(\mu - t)x2_j^{out}(\mu),$$

$$w_{tij}^{out-h}(n) = w_{tij}^{out-h}(n) + \eta(\rho_{tij}^{+} - \rho_{tij}^{-}),\ i \in \overline{1, N^{out}},\ j \in \overline{1, N^h},$$

$$\rho_{tij}^{+} = \frac{1}{P}\sum_{\mu=1}^{P} x1_i^{out}(\mu - t)x1_j^h(\mu),\quad \rho_{tij}^{-} = \frac{1}{P}\sum_{\mu=1}^{P} x2_i^{out}(\mu - t)x2_j^h(\mu),$$

$$w_{ij}^{h-h}(n) = w_{ij}^{h-h}(n) + \eta(\rho_{ij}^{+} - \rho_{ij}^{-}),\ i, j \in \overline{1, N^h},$$

$$\rho_{ij}^{+} = \frac{1}{P}\sum_{\mu=1}^{P} x1_i^h(\mu - t)x1_j^h(\mu),\quad \rho_{ij}^{-} = \frac{1}{P}\sum_{\mu=1}^{P} x2_i^h(\mu - t)x2_j^h(\mu).$$

13. If $\frac{1}{P}\sum_{\mu=1}^{P}\left(\sum_{i=1}^{N^{out}} | x1_i^{out}(\mu) - x2_i^{out}(\mu) |\right) > \varepsilon$, then $n = n + 1$, go to 2.

5. Experiments and results

To determine the structure of the forecasting model based on TDMFBM with 16 input neurons, i.e. determining the amount of hidden neurons, a number of experiments were carried out, the results of which are presented in Figure 3. As input data to determine the values of the parameters of the neural network forecasting model, a sample of values of the economic activities of the logistics company «Ekol Ukraine» was used. The criterion for choosing the structure of the neural network model was the minimum mean squared error (MSE) of forecasting. The dataset capacity for the "cost of transportation" indicator was 1000. The dataset was divided into three parts - training data (60%), test data (20%), test data (20%). The training took place over 100 epochs. The change in the MSE value chosen as the loss function depended on the training epoch number and occurred exponentially. The common parameters for all neural networks were the number of neurons in the hidden layer. As can be seen from Figure 3, with an increase in the amount of hidden neurons the error value decreases. For the forecast, it is sufficient to use 32 hidden neurons, since with a further increase in their amount the change in the error value is insignificant. The work investigated forecasting neural networks according to the criterion of the minimum mean squared error (MSE) of the forecast (Table 2).

According to Table 2, TDMFBM type 2 has the highest forecast accuracy. TDMFBM type 1 can train in burst mode unlike other networks. Thus, type 1 TDMFBM has the fastest learning rate.

Table 2
Comparative characteristics of forecasting neural networks

Network / Criterion	LSTM	GRU	BLSTM	BGRU	ESN	LSM	TDMFBM type 1 / 2
Minimum MSE of the forecast	0.10	0.12	0.09	0.11	0.08	0.10	0.06 / 0.02

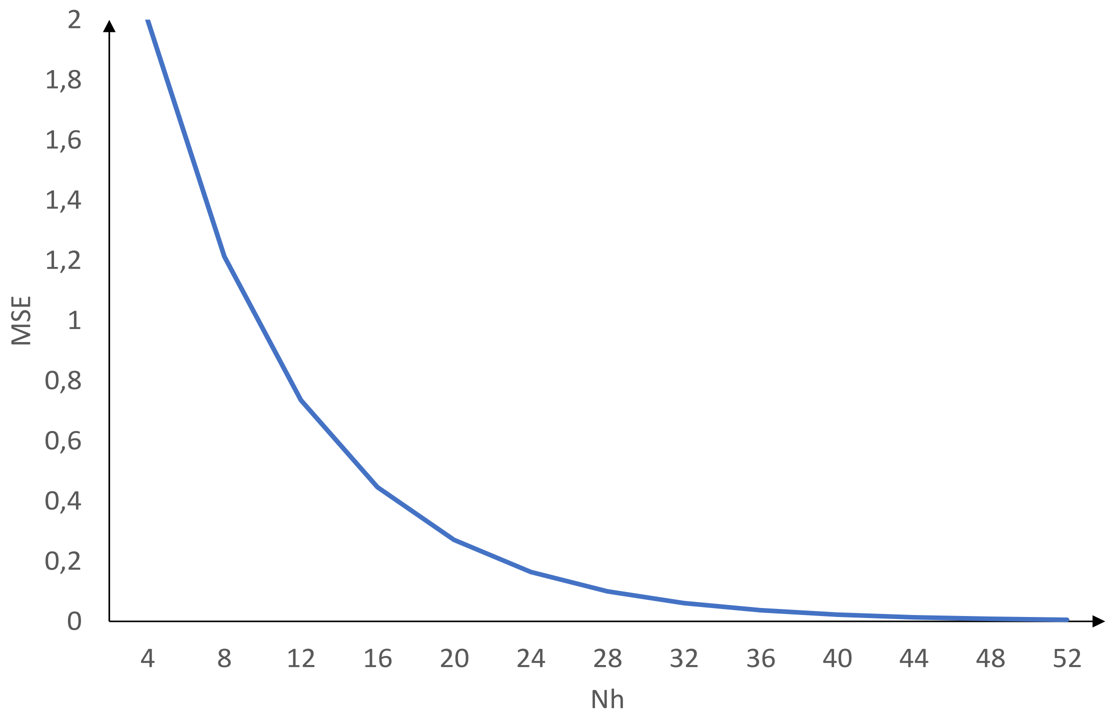

Figure 3: Graph of the dependence of the mean squared error (MSE) value on the amount of hidden neurons

6. Conclusions

1. To solve the problem of improving forecast quality for effective supply chain management, forecast methods were analyzed. According to the studies carried out, neural networks are currently the most effective forecasting tool.

2. In order to improve the forecast efficiency, the MFBM neural network was selected, modified (by introducing time delays in the visible layer), and the structure of its model was identified in the process of numerical study. The conducted study showed that with 32 neurons in the hidden layer, the value of the root mean square error changes little, and the proposed network performs the forecast with a minimum error.

3. A method for calculating the values of the parameters of the created neural network forecast model was proposed. This ensures high accuracy and speed of the forecast.

4. The developed approach can be used for forecasting in various intelligent computer systems of general and special purpose.

7. References

[1] J. F. Cox, J.G. Schleher, Theory of Constraints Handbook, New York, NY, McGraw-Hill, 2010.

[2] S. Smerichevska et al, Cluster Policy of Innovative Development of the National Economy: Integration and Infrastructure Aspects: monograph, S. Smerichevska (Eds.), Wydawnictwo naukowe WSPIA, 2020.

[3] E. M. Goldratt, My saga to improve production, Selected Readings in Constraints Management, Falls Church, VA: APICS (1996) 43-48.

[4] E. M. Goldratt, Production: The TOC Way (Revised Edition) including CD-ROM Simulator and Workbook, Revised edition, Great Barrington, MA: North River Press, 2003.

[5] B. Choubin, G. Zehtabian, A. Azareh, E. Rafiei-Sardooi, F. Sajedi-Hosseini, Ö. Kişi, Precipitation forecasting using classification and regression trees (CART) model: a comparative study of different approaches, Environ Earth Sci 77, 314 (2018). doi: 10.1007/s12665-018-7498-z.

[6] P. Bidyuk, T. Prosyankina-Zharova, O. Terentiev, Modelling nonlinear nonsta-tionary processes in macroeconomy and finances, in: Z. Hu, S. Petoukhov, I. Dychka, M. He (Eds.), Advances in Computer Science for Engineering and Education. Advances in Intelligent Systems and

Computing, volume 754, Springer, Cham, 2019, pp. 735–745. doi: 10.1007/978-3-319-91008-6_72.
[7] R. T. Baillie, G. Kapetanios, F. Papailias, Modified information criteria and selection of long memory time series models, in: Computational Statistics and Data Analysis, volume 76, 2014, pp. 116–131. doi: 10.1016/j.csda.2013.04.012.
[8] L. Lyubchyk, E. Bodyansky, A. Rivtis, Adaptive harmonic components detection and forecasting in wave non-periodic time series using neural networks, in: Proceedings of the ISCDMCI'2002, Evpatoria, 2002, pp. 433-435.
[9] S. N. Sivanandam, S. Sumathi, S. N. Deepa, Introduction to Neural Networks using Matlab 6.0, The McGraw-Hill Comp., Inc., New Delhi, 2006.
[10] A. Wilinski, Time series modelling and forecasting based on a Markov chain with changing transition matrices, in: Expert Systems with Applications, volume 133, 2019, pp. 163–172. doi: 10.1016/j.eswa.2019.04.067.
[11] M. Sundermeyer, R. Schluter, H. Ney, LSTM neural networks for language modeling, in: Thirteenth Annual Conference of the International Speech Communication Association, 2012, pp. 194-197. doi: 10.1.1.248.4448.
[12] P. Potash, A. Romanov, A. Rumshisky, Ghostwriter: using an LSTM for automatic rap lyric generation, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 1919– 1924. doi:10.18653/v1/D15-1221.
[13] E. Kiperwasser, Y. Goldberg, Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations, Transactions of the Association for Computational Linguistics 4 (2016) 313–327. doi: 10.1162/tacl_a_00101.
[14] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures, Neural Networks 18 (2005) 602–610, doi:10.1016/j.neunet.2005.06.042.
[15] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555, 2014.
[16] R. Dey, F. M. Salem, Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks, arXiv:1701.05923, 2017. – URL: https://arxiv.org/ftp/arxiv/papers/1701/1701.05923.pdf.
[17] E. Fedorov, T. Utkina, O. Nechyporenko, Forecast method for natural language constructions based on a modified gated recursive block, in: CEUR Workshop Proceedings, vol. 2604, 2020, pp. 199-214.
[18] Q. Lu,; Z. Zhu, F. Xu, D. Zhang, W. Wu, Q. Guo, Bi-GRU Sentiment Classification for Chinese Based on Grammar Rules and BERT, International Journal of Computational Intelligence Systems 13 (2020) 538-548. doi: 10.2991/ijcis.d.200423.001.
[19] T. Fan, J. Zhu, Y. Cheng, Q. Li, D. Xue, R. Munnoch, A New Direct Heart Sound Segmentation Approach using Bi-directional GRU, in: Proceedings of the 2018 24th International Conference on Automation and Computing, 2018, pp. 1–5. doi: 10.23919/IConAC.2018.8749010.
[20] H. Jaeger, Tutorial on Training Recurrent Neural Networks, Covering BPPT, RTRL, EKF and the Echo State Network Approach, GMD Report 159, German National Research Center for Information Technology, 2002.
[21] H. Jaeger, M. Lukosevicius, D. Popovici, U. Siewert, Optimization and applications of echo state networks with leakyintegrator neurons, in: Neural Networks volume 20, 2007, pp. 335–352. doi:10.1016/j.neunet.2007.04.016.
[22] T. Natshlager, W. Maas, H. Markram, The liquid computer: A novel strategy for real-time computing on time series, in: Special Issue on Foundations of Information Processing of Telematik, 2002, pp. 39–43.
[23] W. Maass, Liquid state machines: motivation, theory, and applications, in: Computability in context: computation and logic in the real world, 2011, pp. 275–296. doi: 10.1142/9781848162778_0008.
[24] T. Neskorodieva, E. Fedorov, I. Izonin, Forecast method for audit data analysis by modified liquid state machine, in: CEUR Workshop Proceedings, 2020, volume 2631, pp. 145-158.
[25] G. E. Hinton, A Practical Guide to Training Restricted Boltzmann Machines, Technical Report UTML TR 2010–003, University of Toronto, 2010.
[26] A. Fischer, C. Igel, Training Restricted Boltzmann Machines: An Introduction, Pattern Recognition 47 (2014) 25-39. doi: 10.1016/j.patcog.2013.05.025.