Neural Network Forecasting Method for Inventory Management in the Supply Chain

Oleg Grygor[1], Eugene Fedorov[1], Olga Nechyporenko[1] and Mykola Grygorian[2]

[1]*Cherkasy State Technological University, Shevchenko blvd., 460, Cherkasy, 18006, Ukraine*
[2]*Cherkasy Institute of Fire Safety named after Chornobyl Heroes of National University of Civil Defence of Ukraine, Onoprienko str., 8, Cherkasy, 18034, Ukraine*

Abstract

Determining the optimal level of inventory comes down to the timeliness of the procurement and replenishment procedures, which ensure the minimum total costs associated with procurement and storage. The problem of insufficient prediction accuracy for inventory management arising in supply chains is considered. A neural network prediction model based on a Time-Delay Restricted Boltzmann Machine with unit delays cascades in the output and input layers is proposed. During the structural identification of this model, the neurons count in the hidden layer was calculated, and the parametric identification was performed based on the CUDA parallel processing technology. This improves the prediction efficiency by increasing the prediction accuracy and decreasing the computational complexity. Software has been developed by the Matlab package that realizes the offered method. The created software is used to implement the prediction in the supply chain management problem.

Keywords

prediction accuracy, supply chain management problem, neural network prediction model, Restricted Boltzmann Machine, stochastic learning

1. Introduction

Despite a large number of studies devoted to the problem of improving the efficiency of supply chains and reducing logistics costs associated with inventory management, some questions remain open. Currently, the problem in the supply chain management (SCM) is not enough high prediction accuracy [1-4]. Thus SCM can be effectless, for example for the concept “Material Requirements Planning” (MRP). Therefore, the development of supply chain prediction methods is an urgent problem.

As of today, many approaches are known as prediction tools, among which are:

- methods based on a modified liquid state machine and a modified gated recurrent unit [5, 6];
- autoregressive and regressive prediction methods [7];
- structural prediction methods Markov chains [8];
- prediction methods exponential smoothing [9];
- logical prediction methods regression and classification trees [10];
- artificial neural network prediction methods [11, 12].

The use of artificial neural networks in prediction provides tangible advantages:

- the relations between features are investigated on ready-made models;
- no guess-work about the features distribution are required;

CMIS-2022: The Fifth International Workshop on Computer Modeling and Intelligent Systems, May 12, 2022, Zaporizhzhia, Ukraine
EMAIL: chdtu-cherkassy@ukr.net (O. Grygor); fedorovee75@ukr.net (E. Fedorov); olne@ukr.net (O. Nechyporenko); hryhorian_mykola@chipb.org.in (M. Grygorian)
ORCID: 0000-0002-5233-290X (O. Grygor); 0000-0003-3841-7373 (E. Fedorov); 0000-0002-3954-3796 (O. Nechyporenko); 0000-0003-0359-5735 (M. Grygorian)

- initial data about the features may be missing;
- source data can be noisy, incomplete or highly correlated;
- systems analysis with a big nonlinearity degree is possible;
- rapid model create;
- high adaptability;
- systems analysis with a big number of features is possible;
- full list of all possible models is not demanded;
- systems analysis with heterogeneous features is possible.

Given the advantages above, the paper will use a artificial neural network prediction method.

The goal of the work is to create a artificial neural network prediction method in the supply chain to improve prediction accuracy. To achieve the aim, the next tasks were solved:

- analyze existing prediction methods;
- propose a artificial neural network prediction model;
- choose a criterion for evaluating the artificial neural network prediction model quality;
- propose a method for calculating parameters of the artificial neural network prediction model;
- conduct a numerical study.

2. Formal problem statement

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

The problem of enhancement the prediction accuracy for the Time-Delay Restricted Boltzmann Machine (TDRBM) model $g(x, W)$, where x – input vector, W – parameters vector, is showed as the problem of searching for this model such a parameters vector W^* that meets the criterion $F = \frac{1}{P}\sum_{\mu=1}^{P}(g(x_\mu, W^*) - d_\mu)^2 \to \min$.

3. Literature review

The often used prediction artificial neural networks are:

- Jordon's neural network (JNN) [13, 14]. It is a recurrent two-layer network and is founded on multilayer perceptron (MLP). JNN is recommended for prediction stationary short-term signals;
- Elman's neural network (ENN) [15, 16]. It is a recurrent two-layer network and is founded on MLP. SRN is recommended for prediction stationary short-term signals;
- recurrent multilayer perceptron (RMLP) [17, 18]. It is a recurrent multilayer network and is founded on MLP. RMLP is recommended for prediction stationary short-term signals;
- nonlinear autoregressive model (NAR) [19, 20]. It is a non-recurrent two-layer network and is founded on MLP. NAR is recommended for prediction stationary short-term signals;
- nonlinear autoregressive moving average model (NARMA) [21, 22]. It is a recurrent two-layer network and is founded of MLP. NARMA is recommended for prediction stationary short-term signals;
- bidirectional recurrent neural network (BRNN) [23, 24]. It is a recurrent two-layer network and is founded on two Elman networks. BRNN is recommended for prediction non-stationary long-term signals.

Table 1 shows the comparative features of prediction artificial neural networks.

As follows from Table 1, most neural networks have one or more of the following disadvantages:

- have a high probability of hitting a local extremum;
- do not have the ability to train in batch mode;
- have no feedback.

Due to this, it is relevant to create a neural network that will eliminate the indicated disadvantages.

Table 1
Comparative features of prediction artificial neural networks

Criterion \ Network	RMLP	JNN	ENN (SRN)	NAR	NARMA	BRNN
The presence of feedback	+	+	-	-	+	+
Unit delay cascade	-	-	-	+	+	-
Low probability of local extremum	-	-	-	-	-	-
High training speed	+	+	+	+	+	+
Possibility of batch learning	-	-	-	+	-	-

4. Block diagram of the neural network prediction model

Figures 1 and 2 show the block diagrams of a prediction model based on a Time-Delay Restricted Boltzmann Machine (TDRBM) of two types, which are stochastic recurrent neural networks with one hidden layer.

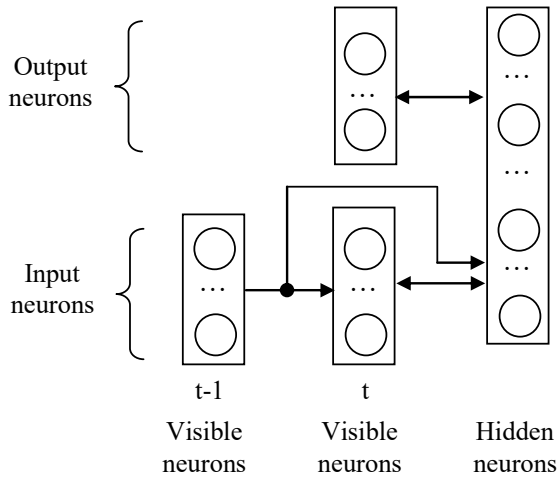

Figure 1: Block diagram of a prediction model TDRBM for visible input neurons (TDRBM type first)

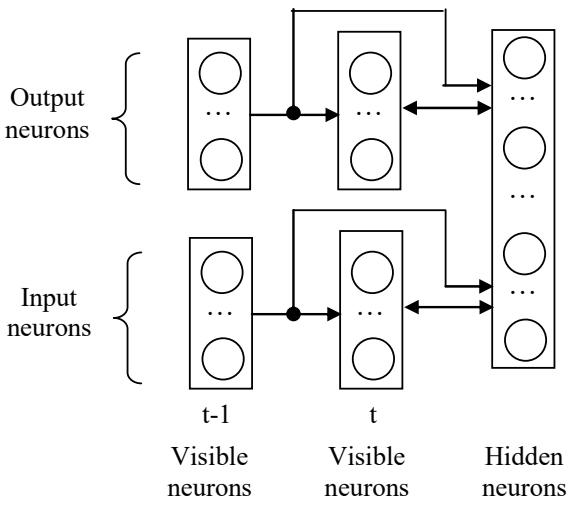

Figure 2: Block diagram of a prediction model TDRBM for the visible input neurons and visible output neurons (TDRBM type second)

Unlike the traditional Restricted Boltzmann Machine (RBM) [25, 26], unit delay cascades are used for neurons in the visible layer. TDRBM type first has unit delay cascade in the visible input layer. TDRBM type seconf has unit delay cascade in the visible input layers and visible output layers.

5. Neural network prediction models

5.1. Components of the prediction model TDRBM

The components of the TDRBM model are stochastic neurons, the state of which is described based on the Bernoulli distribution in the form

$$x_j = \begin{cases} 1, & \text{with probability } P_j \\ 0, & \text{with probability } 1 - P_j \end{cases}.$$

The probability of transition of the j th stochastic neuron to state 1 is defined as

$$P_j = \frac{1}{1 + \exp\left(-\Delta E_j\right)},$$

where ΔE_j – is the TDRBM energy increment when the state of the j th stochastic neuron changes from 0 to 1.

5.2. Prediction model TDRBM type first

Positive phase (steps 1-4)

1. Initial value assignment time moment

$$\mu = 1.$$

Initial value assignment of the time delay input neurons state

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

2. Initial value assignment of the input and output neurons state

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

3. Calculation of the hidden neurons transition probability to state 1

$$P_j = \frac{1}{1 + \exp\left(-b_j^h - \sum_{t=0}^{M^{in}} \sum_{i=1}^{N^{in}} w_{tij}^{in-h} x_i^{in}(\mu - t) - \sum_{i=1}^{N^{out}} w_{ij}^{out-h} x_i^{out}(\mu)\right)},\ j \in \overline{1, N^h}$$

where w_{tij}^{in-h} – weights between the input layer and the hidden layer,

w_{ij}^{out-h} – weights between the output layer and the hidden layer,

b_j^h – bias of the hidden layer neurons,

M^{in} – the unit delays count for the input layer,

N^{in} – the neurons count in the input layer,

N^h – the neurons count in the hidden layer,

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

4. Calculation of the hidden neurons state

$$x_j^h(\mu) = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases},\ j \in \overline{1, N^h}$$

where $U(0,1)$ – standard uniform distribution on a line $[0,1]$.

Negative phase (step 5)

5. Calculation of the output neurons transition probability to state 1

$$P_j = \frac{1}{1+\exp\left(-b_j^{out} - \sum_{i=1}^{N^h} w_{ij}^{out-h} x_i^h(\mu)\right)},\ j \in \overline{1, N^{out}}$$

where b_j^{out} – bias of the output layer neurons.

6. Calculation of the output neurons state

$$x_j^{out}(\mu) = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases},\ j \in \overline{1, N^{out}}.$$

The outcome is $(x_1^{out}(\mu), ..., x_{N^{out}}^{out}(\mu))$.

5.3. Prediction model based on TDRBM type second

Positive phase (steps 1-4)

1. Initial value assignment time moment

$$\mu = 1.$$

Initial value assignment time moment of the time delay input and output neurons state

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

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

2. Initial value assignment of the input and output neurons state

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

3. Calculation of the hidden neurons transition probability to state 1

$$P_j = \frac{1}{1+\exp\left(-b_j^h - \sum_{t=0}^{M^{in}} \sum_{i=1}^{N^{in}} w_{tij}^{in-h} x_i^{in}(\mu - t) - \sum_{t=0}^{M^{out}} \sum_{i=1}^{N^{out}} w_{tij}^{out-h} x_i^{out}(\mu - t)\right)},\ j \in \overline{1, N^h},$$

where w_{tij}^{in-h} –weights between the input layer and the hidden layer,

w_{tij}^{out-h} – weights between the output layer and the hidden layer,

b_j^h – bias of the hidden layer neurons,

M^{in} – the unit delays count for the input layer,

M^{out} – the unit delays count for the output layer,

N^{in} – the neurons count in the input layer,

N^h – the neurons count in the hidden layer,

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

4. Computation of the hidden neurons state

$$x_j^h(\mu) = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases},\ j \in \overline{1, N^h}$$

where $U(0,1)$ – standard uniform distribution on a line $[0,1]$.

Negative phase (step 5)

5. Calculation of the output neurons transition probability to state 1

$$P_j = \frac{1}{1+\exp\left(-b_j^{out} - \sum_{t=1}^{M^{out}} \sum_{i=1}^{N^{out}} w_{tij}^{out-out} x_i^{out}(\mu - t) - \sum_{i=1}^{N^h} w_{0ij}^{out-h} x_i^h(\mu)\right)},\ j \in \overline{1, N^{out}}$$

6. Calculation of the output neurons state

$$x_j^{out}(\mu)=\begin{cases}1, & P_j \ge U(0,1)\\ 0, & P_j < U(0,1)\end{cases},\ j\in\overline{1,N^{out}}.$$

The outcome is $(x_1^{out}(\mu),...,x_{N^{out}}^{out}(\mu))$.

6. Criterion for the artificial neural network prediction model quality

For learning the TDRBM, a model adequacy criterion was selection, which means the selection of such parameters $W=\{w_{tij}^{in-h},\ w_{tij}^{out-h},\ w_{tij}^{in-in},\ w_{tij}^{out-out}\}$, which get a mean squared error (MSE) minimum:

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

The learning of the TDRBM model is taking into account the criterion (1).

7. Methods for calculating the neural network prediction model parameters

7.1. Principles of calculating the neural network prediction model parameters

The determination of the TDRBM parameters is perform on the found of the CD-1 method, which speeds up stochastic learning with a teacher, since instead of stabilizing the state of neurons, it performs only one step of adjusting their state. TDRBM operates in two phases: positive and negative.

For *TDRBM type first*, in the positive phase, visible input neurons are fixed at times from $t-M^{in}$ to t and visible output neurons at time t, and the TDRBM performs one step of tuning hidden neurons. In the negative phase, the visible input neurons are first fixed at times from $t-M^{in}$ to $t-1$, the hidden neurons are trained in the positive phase, and the TDRBM performs one step of tuning the visible input and output neurons at time t. After that, the visible input neurons are fixed at times from $t-1$ to $t-M^{in}$, and the visible input and output neurons trained in the negative phase at time t, and the TDRBM performs one step of tuning the hidden neurons.

For *TDRBM type second*, in the positive phase visible input neurons are fixed at times $t-M^{in}$ to t, visible output neurons at times from $t-M^{out}$ to t, and the TDRBM performs one step of tuning hidden neurons. In the negative phase, the visible input neurons are first fixed at times from $t-M^{in}$ to $t-1$, the visible output neurons at times from $t-M^{out}$ to $t-1$, the hidden neurons are trained in the positive phase, and the TDRBM performs one step of tuning the visible input and output neurons at time t. After that, visible input neurons are recorded at times from $t-1$ to $t-M^{in}$, visible output neurons at times from $t-1$ to $t-M^{out}$, and visible input and output neurons trained in the negative phase at time t, and TDRBM performs one step of tuning hidden neurons.

7.2. Method for calculating the prediction model parameters found on TDRBM type first

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

$t \in \overline{1, M^{in}}$, $i, j \in \overline{1, N^{in}}$, $w_{tii}^{in-h}(n) = 0$, $w_{ii}^{out-h}(n) = 0$, $w_{tii}^{in-in}(n) = 0$, $w_{0ij}^{in-h}(n) = w_{0ji}^{in-h}(n)$, $w_{ij}^{out-h}(n) = w_{ji}^{out-h}(n)$, where M^{in} is the unit delays count for input neurons.

2. A learning 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 defined, where $\mathbf{x}_{\mu}^{in}$ – μ^{th} learning input vector, $\mathbf{x}_{\mu}^{out}$ – μ^{th} learning output vector, P is the learning set capacity.

Positive phase (steps 3-7)

3. Initial value assignment time moment

$$\mu = 1.$$

Initial value assignment of the time delay input neurons state

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

4. Initial value assignment of the input and output neurons state

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

5. Calculation of the hidden neurons transition probability to state 1

$$P_j = \frac{1}{1 + \exp\left(-b_j^h(n) - \sum_{t=0}^{M^{in}} \sum_{i=1}^{N^{in}} w_{tij}^{in-h}(n) x_i^{in}(\mu - t) - \sum_{i=1}^{N^{out}} w_{ij}^{out-h}(n) x_i^{out}(\mu)\right)},\ j \in \overline{1, N^h}$$

6. Calculation of the hidden neurons state

$$x_j^h(\mu) = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases},\ j \in \overline{1, N^h}$$

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

Negative phase (steps 8-16)

8. Initial value assignment time moment

$$\mu = 1.$$

Initial value assignment of the time delay input neurons state

$$\mathbf{x}^{in}(\mu - t) = 0,\ \mathbf{x}2^{in}(\mu - t) = 0,\ t \in \overline{1, M^{in}}.$$

9. Initial value assignment of the of hidden, input and output neurons state

$$\mathbf{x}^{in}(\mu) = \mathbf{x}1^{in}(\mu),\ \mathbf{x}^{out}(\mu) = \mathbf{x}1^{out}(\mu),\ \mathbf{x}^{h}(\mu) = \mathbf{x}1^{h}(\mu).$$

10. Calculation of the output neurons transition probability to state 1

$$P_j = \frac{1}{1 + \exp\left(-b_j^{out}(n) - \sum_{i=1}^{N^h} w_{ij}^{out-h}(n) x_i^h(\mu)\right)},\ j \in \overline{1, N^{out}}.$$

11. Calculation of the of output neurons state

$$x_j^{out}(\mu) = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases},\ j \in \overline{1, N^{out}}.$$

12. Calculation of the input neurons transition probability to state 1

$$P_j = \frac{1}{1 + \exp\left(-b_j^{in}(n) - \sum_{t=1}^{M^{in}} \sum_{i=1}^{N^{in}} w_{tij}^{in-in}(n) x_i^{in}(\mu - t) - \sum_{i=1}^{N^h} w_{0ij}^{in-h}(n) x_i^h(\mu)\right)},\ j \in \overline{1, N^h}.$$

13. Calculation of the input neurons state

$$x_j^{in}(\mu)=\begin{cases}1, & P_j \ge U(0,1)\\ 0, & P_j < U(0,1)\end{cases},\ j\in\overline{1,N^h}.$$

14. Calculation of the hidden neurons transition probability to state 1

$$P_j=\frac{1}{1+\exp\left(-b_j^h(n)-\sum_{t=0}^{M^{in}}\sum_{i=1}^{N^{in}}w_{tij}^{in-h}(n)x_i^{in}(\mu-t)-\sum_{i=1}^{N^{out}}w_{ij}^{out-h}(n)x_i^{out}(\mu)\right)},\ j\in\overline{1,N^h}.$$

15. Calculation of the hidden neurons state

$$x_j^h(\mu)=\begin{cases}1, & P_j \ge U(0,1)\\ 0, & P_j < U(0,1)\end{cases},\ j\in\overline{1,N^h}.$$

16. Saving the neurons state in the negative phase, 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 9.

17. Tuning of bias input layer founded on Boltzmann's rule

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

18. Tuning of bias output layer founded on Boltzmann's rule

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

19. Tuning of bias hidden layer neurons founded on Boltzmann's rule

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

20. Tuning of weights between the input layer and the hidden layer founded on Boltzmann's rule

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

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

$$\varphi(x_i,x_j)=\begin{cases}1, & x_i=x_j\\ 0, & x_i\ne x_j\end{cases}.$$

21. Tuning of weights between the output layer and the hidden layer founded on Boltzmann's rule

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

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

22. Tuning of weights between input layers founded on Boltzmann's rule

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

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

23. If the neurons states in the output layer of the positive and negative phases differ slightly, i.e. $\frac{1}{P}\sum_{\mu=1}^{P}\left(\sum_{i=1}^{N^{out}}|x1_i^{out}(\mu)-x2_i^{out}(\mu)|\right)>\varepsilon$, then $n=n+1$, go to 2.

The method for calculating the prediction model parameter based on TDRBM type first is shown in Figure 3.

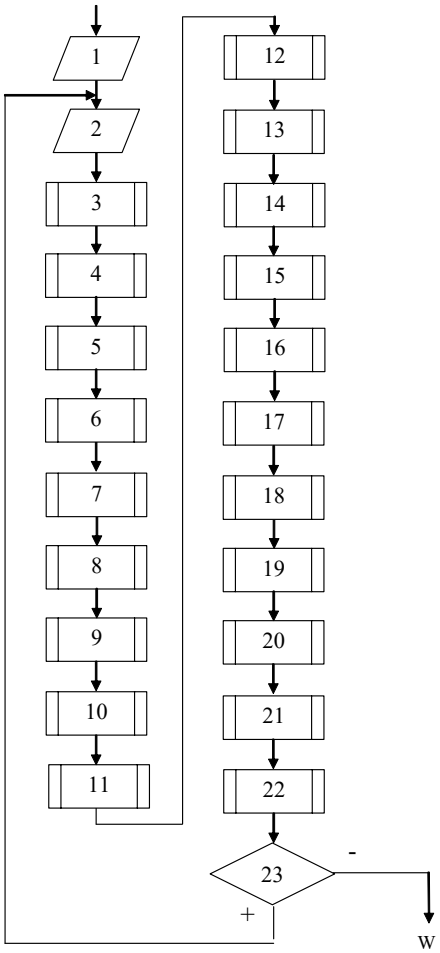

Figure 3: Flowchart of the method for calculating the prediction model parameters found on TDRBM type first

7.3. Method for calculating the prediction model parameters found on TDRBM type second

1. Learning iteration number $n = 1$, initial value assignment by uniform distribution means on the interval (0,1) or [-0.5, 0.5] bias $b_i^{in}(n)$, $i \in \overline{1, N^{in}}$, $b_i^{out}(n)$, $i \in \overline{1, N^{out}}$, $b_j^h(n)$, $j \in \overline{1, N^h}$, and weights $w_{tij}^{in-h}(n)$, $t \in \overline{0, M^{in}}$, $i \in \overline{1, N^{in}}$, $j \in \overline{1, N^h}$, $w_{tij}^{out-h}(n)$, $t \in \overline{0, M^{out}}$, $i \in \overline{1, N^{out}}$, $j \in \overline{1, N^h}$, $w_{tij}^{in-in}(n)$, $t \in \overline{1, M^{in}}$, $i, j \in \overline{1, N^{in}}$, $w_{tij}^{out-out}(n)$, $t \in \overline{1, M^{out}}$, $i, j \in \overline{1, N^{out}}$, $w_{tii}^{in-h}(n) = 0$, $w_{ii}^{out-h}(n) = 0$, $w_{tii}^{in-in}(n) = 0$, $w_{0ij}^{in-h}(n) = w_{0ji}^{in-h}(n)$, $w_{ij}^{out-h}(n) = w_{ji}^{out-h}(n)$, where M^{in} is the unit delays count for input neurons, M^{out} is the unit delays count for output neurons.
2. A learning 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 defined, $\mu \in \overline{1, P}$, where $\mathbf{x}_\mu^{in}$ – μ^{th} learning input vector, $\mathbf{x}_\mu^{out}$ – μ^{th} learning output vector, P is the learning set capacity.

Positive phase (steps 3-7)

3. Initial value assignment time moment

$$\mu = 1 .$$

Initial value assignment of the time delay input and output neurons state

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

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

4. Initial value assignment of the input and output neurons state

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

5. Calculation of the hidden neurons transition probability to state 1

$$P_j=\frac{1}{1+\exp\left(-b_j^h(n)-\sum_{t=0}^{M^{in}}\sum_{i=1}^{N^{in}}w_{tij}^{in-h}(n)x_i^{in}(\mu-t)-\sum_{t=0}^{M^{out}}\sum_{i=1}^{N^{out}}w_{tij}^{out-h}(n)x_i^{out}(\mu-t)\right)},\ j\in\overline{1,N^h}.$$

6. Calculation of the hidden neurons state

$$x_j^h(\mu)=\begin{cases}1, & P_j\geq U(0,1)\\ 0, & P_j<U(0,1)\end{cases},\ j\in\overline{1,N^h}.$$

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

Negative phase (steps 8-16)

8. Initial value assignment time moment

$$\mu=1.$$

Initial value assignment of the time delay input and output neurons state

$$\mathbf{x}^{in}(\mu-t)=0,\ \mathbf{x}2^{in}(\mu-t)=0,\ t\in\overline{1,M^{in}},$$

$$\mathbf{x}^{out}(\mu-t)=0,\ \mathbf{x}2^{out}(\mu-t)=0,\ t\in\overline{1,M^{out}}.$$

9. Initial value assignment of the of hidden, input and output neurons state

$$\mathbf{x}^{in}(\mu)=\mathbf{x}1^{in}(\mu),\ \mathbf{x}^{out}(\mu)=\mathbf{x}1^{out}(\mu),\ \mathbf{x}^{h}(\mu)=\mathbf{x}1^{h}(\mu).$$

10. Calculation of the output neurons transition probability to state 1

$$P_j=\frac{1}{1+\exp\left(-b_j^{out}(n)-\sum_{t=1}^{M^{out}}\sum_{i=1}^{N^{out}}w_{tij}^{out-out}(n)x_i^{out}(\mu-t)-\sum_{i=1}^{N^h}w_{0ij}^{out-h}(n)x_i^{h}(\mu)\right)},\ j\in\overline{1,N^{out}}.$$

11. Calculation of the of output neurons state

$$x_j^{out}(\mu)=\begin{cases}1, & P_j\geq U(0,1)\\ 0, & P_j<U(0,1)\end{cases},\ j\in\overline{1,N^{out}}.$$

12. Calculation of the input neurons transition probability to state 1

$$P_j=\frac{1}{1+\exp\left(-b_j^{in}(n)-\sum_{t=1}^{M^{in}}\sum_{i=1}^{N^{in}}w_{tij}^{in-in}(n)x_i^{in}(\mu-t)-\sum_{i=1}^{N^h}w_{0ij}^{in-h}(n)x_i^{h}(\mu)\right)},\ j\in\overline{1,N^{in}}.$$

13. Calculation of the input neurons state

$$x_j^{in}(\mu)=\begin{cases}1, & P_j\geq U(0,1)\\ 0, & P_j<U(0,1)\end{cases},\ j\in\overline{1,N^{in}}.$$

14. Calculation of the hidden neurons transition probability to state 1

$$P_j=\frac{1}{1+\exp\left(-b_j^h(n)-\sum_{t=0}^{M^{in}}\sum_{i=1}^{N^{in}}w_{tij}^{in-h}(n)x_i^{in}(\mu-t)-\sum_{t=0}^{M^{out}}\sum_{i=1}^{N^{out}}w_{tij}^{out-h}(n)x_i^{out}(\mu-t)\right)},\ j\in\overline{1,N^h}.$$

15. Calculation of the hidden neurons state

$$x_j^h(\mu)=\begin{cases}1, & P_j\geq U(0,1)\\ 0, & P_j<U(0,1)\end{cases},\ j\in\overline{1,N^h}.$$

16. Saving the neurons state in the negative phase, 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 9.
17. Tuning of bias input layer founded on Boltzmann's rule

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

18. Tuning of bias output layer founded on Boltzmann's rule

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

19. Tuning of bias hidden layer neurons founded on Boltzmann's rule

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

20. Tuning of weights between the input layer and the hidden layer founded on Boltzmann's rule

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

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

$$\varphi(x_i,x_j)=\begin{cases}1, & x_i=x_j\\ 0, & x_i\neq x_j\end{cases}.$$

21. Tuning of weights between the output layer and the hidden layer founded on Boltzmann's rule

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

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

22. Tuning of weights between input layers founded on Boltzmann's rule

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

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

23. Tuning of weights between output layers founded on Boltzmann's rule

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

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

24. If the neurons states in the output layer of the positive and negative phases differ slightly, i.e. $\frac{1}{P}\sum_{\mu=1}^{P}\left(\sum_{i=1}^{N^{out}}|x1_i^{out}(\mu)-x2_i^{out}(\mu)|\right)>\varepsilon$, then $n=n+1$, go to 2.

The method for calculating the prediction model parameters found on TDRBM type second is shown in Figure 4.

8. Experiments and results

A numerical study of the proposed method for determining the parameter values was carried out using the CUDA technology of parallel processing of information in the Matlab package, wherein the number of threads in the block was 1024, the summation was performed based on the parallel reduction algorithm.

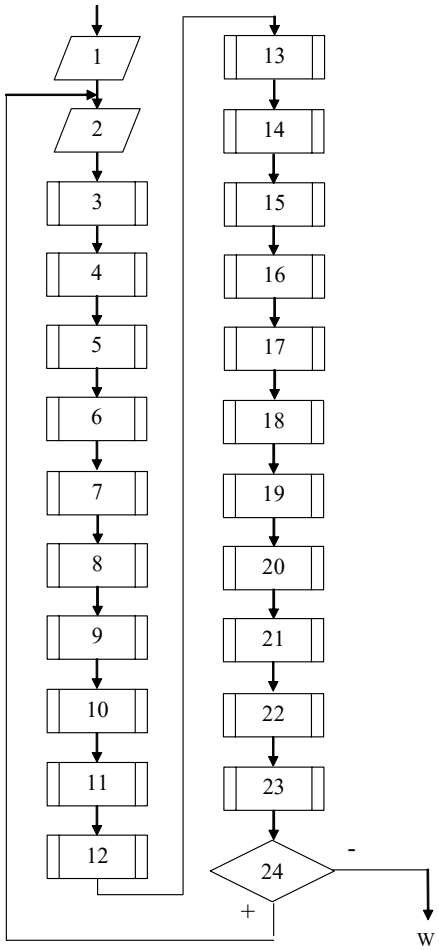

Figure 4: Flowchart of the method for calculating the prediction model parameters found on TDRBM type second

To determine the prediction model structure TDRBM with 16 input neurons (the bits count of the integer predicted metric), i.e. defining the number of hidden neurons, a experiments count were performed, the results of which are showed in Figure 5.

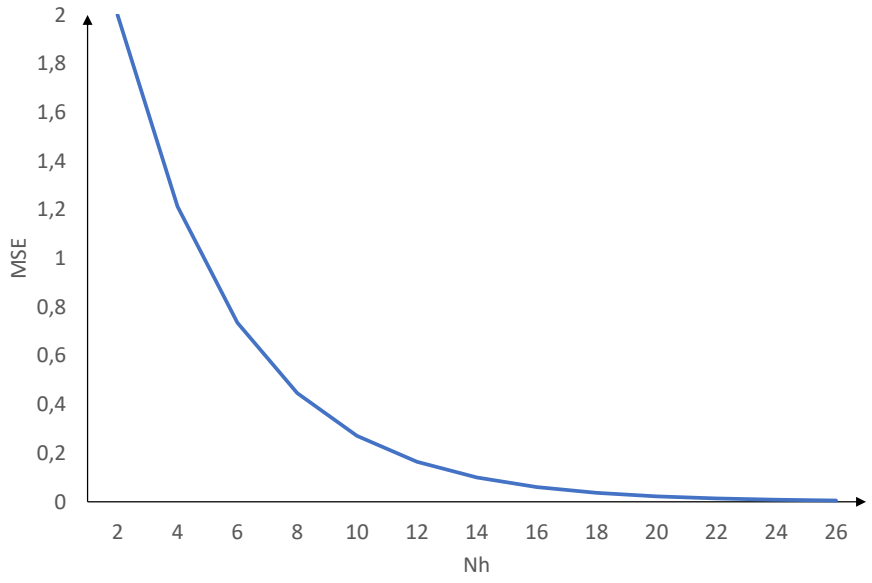

Figure 5: The mean squared error dependence on the hidden neurons number

As source data to calculate the artificial neural network prediction model parameters, a values sample of the logistics company "Ekol Ukraine" economic activities was used. The dataset capacity for the "goods sold" indicator was 1800 (in five years). The dataset was spilt into three parts - valid data (20%), training data (60%), test data (20%). The learning was consistent and occurred over 1000 epochs.

The criterion for selection the artificial neural network model structure of the was the prediction minimum mean squared error (MSE) (Table 2). According to Figure 5, with an increase in the hidden neurons count the error value decreases. For the prediction, it is enough to choose 16 hidden neurons, because with a further increase in their count the change in the MSE value is small.

Table 2

Comparative features of prediction artificial neural networks

Criterion \ Network	RMLP	JNN	ENN (SRN)	NAR	NARMA	BRNN	TDRBM type first / second
Minimum MSE of the prediction	0.17	0.12	0.10	0.15	0.07	0.05	0.04 / 0.02

According to Table 2, recurrent neural networks with deterministic neurons (JNN, ENN(SRN), NARMA, BRNN) are more efficient than non-recurrent neural networks with deterministic neurons (RMLP, NAR); neural networks with stochastic neurons (TDRBM) are more efficient than neural networks with deterministic neurons (RMLP, JNN, ENN(SRN), NAR, NARMA, BRNN), and TDRBM type second has the highest prediction accuracy.

9. Conclusion

1. To solve the problem of enhancement the prediction accuracy in the supply chain, the modern prediction methods were researched. These researches have shown that today the use of neural networks is the most effective.
2. To improve the efficiency of the prediction, a artificial neural network RBM was modified by the unit delays cascades in the input and output layers. In the experiments, its model structure was determined.
3. A method was offered for calculating the parameters of the offered artificial neural network prediction model based on the CUDA technology of parallel processing of information. This made it possible to ensure accuracy of the prediction and high speed.
4. The offered approach can be used in different intelligent prediction systems. For example, in supply chain inventory management systems such as MRP and Lean Production, where prediction plays an important role.

10. 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] 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.

[6] 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.
[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] 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.
[9] P. Bidyuk, T. Prosyankina-Zharova, O. Terentiev, Modelling nonlinear nonstationary processes in macroeconomy and finances, in: Advances in Computer Science for Engineering and Education, volume 754, Springer, 2019, pp. 735–745. doi: 10.1007/978-3-319-91008-6_72.
[10] 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 Sciences 77, 314 (2018). doi: 10.1007/s12665-018-7498-z.
[11] 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.
[12] S. N. Sivanandam, S. Sumathi, S. N. Deepa, Introduction to Neural Networks using Matlab 6.0, The McGraw-Hill Comp., Inc., New Delhi, 2006.
[13] E. C. Carcano, P. Bartolini, M. Muselli and L. Piroddi, Jordan recurrent neural network versus ihacres in modelling daily streamflows, Journal of Hydrology 362 (2008) 291-307.
[14] H. Hikawa and Y. Araga, Study on gesture recognition system using posture classifier and Jordan recurrent neural network, in: Proceedings of the International Joint Conference on Neural Networks, 2011, pp. 405-412. doi: 10.1109/IJCNN.2011.6033250.
[15] Z. Zhang, Z. Tang, C. Vairappan, A novel learning method for Elman neural network using local search, in: Neural Information Processing – Letters and Reviews, vol. 11, 2007, pp. 181–188.
[16] A. Wysocki and M. Ławryńczuk, Predictive control of a multivariable neutralisation process using Elman neural networks, in: Advances in Intelligent Systems and Computing, Springer: Heidelberg, 2015, pp. 335-344. doi: 10.1007/978-3-319-15796-234.
[17] S. Haykin, Neural networks and Learning Machines, Upper Saddle River, New Jersey: Pearson Education, Inc., 2009.
[18] K.-L. Du, K. M. S. Swamy, Neural Networks and Statistical Learning, Springer-Verlag, London, 2014.
[19] J. A. Pucheta, C. M. Rodríguez Rivero, M. R. Herrera, C. A. Salas, H. D. Patiño and B. R. Kuchen, A feed-forward neural networks-based nonlinear autoregressive model for forecasting time series, Computación y Sistemas 14 (4) (2011) 423-435.
[20] G. P. Zhang, Time series forecasting using a hybrid ARIMA and neural network model, Neurocomputing 50 (2003) 159–175. doi: 10.1016/S0925-2312(01)00702-0.
[21] W.K. Wong, M. Xia, W. C. Chu, Adaptive neural network model for time-series forecasting, European Journal of Operational Research 207(2) (2010) 807-816. doi: 10.1016/j.ejor.2010.05.022.
[22] H. Jaeger, H. Haass, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science 304 (2004) 78–80. doi: 10.1126/science.1091277.
[23] M. Sundermeyer, T. Alkhouli, J. Wuebker, H. Ney, Translation modeling with bidirectional recurrent neural networks, in: Proceedings of the Conference on Empirical Methods on Natural Language Processing, 2014, pp. 14-25.
[24] M. Berglund, T. Raiko, M. Honkala, L. Kärkkäinen, A. Vetek, J. Karhunen, Bidirectional Recurrent Neural Networks as Generative Models, in: Proceedings of the 28th International Conference on Neural Information Processing Systems, 2015, pp. 856−864.
[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.