METHOD FOR AUTOMATIC PROCESSING OF AUDIT CONTENT BASED ON BIDIRECTIONAL NEURAL NETWORK MAPPING

Tatiana Neskorodieva[a], Eugene Fedorov[a,b]

[a] *Vasyl' Stus Donetsk National University, 600-richchia str., 21, Vinnytsia, 21021, Ukraine*
[b] *Cherkasy State Technological University, Shevchenko blvd., 460, Cherkasy, 18006, Ukraine*

Abstract

Currently, the analytical procedures used during the audit are based on data mining techniques. The work solves the problem of increasing the efficiency and effectiveness of analytical audit procedures by automating data comparison by bidirectional neural network mapping.

The object of the research is the process of the content auditing of the receipt of raw materials for production and the manufactured products.

The aim of the work is to increase the effectiveness and efficiency of audit due to mapping by full (bidirectional) counterpropagating neural network of content of the receipt of raw materials for production and the manufactured products while automating procedures for checking their compliance.

The vectors of feature for the objects of the sequences of the receipt of raw materials for production and the manufactured products are generated, which are then used in the proposed method. The created method, in contrast to the traditional one, provides for a batch mode, which allows the method to increase the learning rate by an amount equal to the product of the number of neurons in the hidden layer and the power of the training set, which is critically important in the audit system for the implementation of multivariate intelligent analysis, which involves enumerating various methods of forming subsets analysis.

The urgent task of increasing the audit efficiency was solved by automating the mapping of audit indicators by full (bidirectional) counterpropagating neural network. A learning algorithm based on k-means has been created, intended for implementation on a GPU using CUDA technology, which increases the speed of identifying parameters of a neural network model.

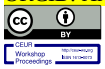
The neural network with the proposed training method based on the k-means rule can be used to intellectualize the DSS audit. The prospects for further research are the application of the proposed method by neural network mapping for a wide class of artificial intelligence tasks, in particular, for creating a method for bidirectional mapping indicators of audit tasks.

Keywords[1]

audit, mapping by neural network, full (bidirectional) counterpropagating neural network, content of the receipt of raw materials for production and the manufactured products.

1. Introduction

In the process of development of international and national economies and industry of IT in particular, it is possible to distinguish the following basic tendencies: realization of digital transformations, forming of digital economy, globalization of socio-economic processes and of IT accompanying them [1]. These processes result in the origin of global, multilevel hierarchical structures of heterogeneous, multivariable, multifunction connections, interactions and cooperation of managing subjects (objects of audit), the large volumes of information about them have been accumulated in the informative systems of account, management and audit.

Proccedings Name, Month XX–XX, YYYY, City, Country
EMAIL: email1@mail.com (A. 1); email2@mail.com (A. 2); email3@mail.com (A. 3)
ORCID: XXXX-XXXX-XXXX-XXXX (A. 1); XXXX-XXXX-XXXX-XXXX (A. 2); XXXX-XXXX-XXXX-XXXX (A. 3)

© 2020 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

Consequently, nowadays the scientific and technical issue of the modern information technologies in financial and economic sphere of Ukraine is forming of the methodology of planning and creation of the decision support systems (DSS) at the audit of enterprises in the conditions of application of IT and with the use of information technologies on the basis of the automated analysis of the large volumes of data about financial and economic activity and states of enterprises with the multi-level hierarchical structure of heterogeneous, multivariable, multifunction connections, intercommunications and cooperation of objects of audit with the purpose of expansion of functional possibilities, increase of efficiency and universality of IT-audit.

Automated DSS audit means the automatic forming of recommendable decisions, based on the results of the automated analysis of data, that improves quality process of audit. Unlike the traditional approach, computer technologies of analysis of data in the system of audit accelerate and promote the process accuracy of audit, that extremely critical in the conditions of plenty of associate tasks on lower and middle levels, and also amounts of indexes and supervisions in every task.

When developing a decision-making system in audit based on data mining technologies, three methods have been created: classifying variables, forming analysis sets, mapping analysis sets [2,3].

The peculiarity of the methodology for classifying indicators is that qualitatively different (by semantic content) variables are classified: numerological, linguistic, quantitative, logical. The essence of the second technique is determined by the qualitative meaning of the indicators. In accordance with this, sets are formed with the corresponding semantic content: document numbers, the name of indicators, quantitative estimates of the values of indicators, logical indicators.

The third technique is subordinated to the mappings of formed sets of the same type on each other in order to determine equivalence in the following senses: numerological, linguistic, quantitative, logical.

The most urgent problem is the mapping of quantitative indicators. The mapping of quantitative indicators of the audit can be implemented through an ANN with associative memory. The main ANNs with associative memory are presented in Table 1. The memory capacity was considered only for ANNs with a binary or bipolar data type that perform reconstruction or classification. HAM stands for hetero-associative memory, AAM stands for auto-associative memory.

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

1. not used to reconstruct either the original or another sample;
2. do not work with real data.
3. do not have a high capacity of associative memory.

The aim of the work is to increase the efficiency of automatic data analysis in the audit DSS by means of a bidirectional (forward and reverse) neural network mapping of sets of audit indicators in order to identify systematic misstatements that lead to misstatement of reporting. In the audit system, the topical task of the middle level is the automation of the analysis of the conformity of the content of the supply of raw materials for production and the manufactured products (by the periods of quantization of the verification period).

It is assumed that the audit indicators are noisy with Gaussian noise, which in turn simulates random accounting errors (as opposed to systematic ones). It is also assumed that the residuals for the quantization periods are distributed according to the normal law, the parameters of which can be estimated from the accounting data. For the achievement of the aim it is necessary to solve the following tasks:

- formalize the content of the audit process of the receipt of raw materials for production and the manufactured products;
- choose a neural network model for mapping audit indicators (which are noisy with Gaussian noise, which in turn simulates random accounting errors (as opposed to systematic ones, which lead to distortion of reporting));
- choose a criterion for evaluating the effectiveness of a neural network model;
- propose a method for training a neural network model in batch mode;
- propose an algorithm for training a neural network model in batch mode for implementation on a GPU;
- perform numerical studies.

Table 1
Basic ANNs with associative memory

ANN	Memory type	Memory capacity	Data type	Purpose
Forward-only Counterpropagation Neural Network [4, 5]	HAM	-	Real	Reconstruction of other sample
Full (bi-directional) Counterpropagation Neural Network [7]	AAM, HAM	-	Real	Reconstruction of the original or other sample
Deep Belief Network [8]	AAM	Medium	Binary	Reconstruction of the original sample
Restricted Boltzmann machine [9, 10]	AAM	Medium	Binary	Reconstruction of the original sample
Self-Organizing map [11, 12]	AAM	-	Real	Clustering
Learning Vector Quantization [13]	AAM	-	Real	Clustering
Principal Component Analysis NN [14]	HAM	-	Real	Dimension reduction
Independent Component Analysis NN [15, 16]	HAM	-	Real	Dimension reduction
Cerebellar Model Articulation Controller [17]	HAM	-	Real	Coding
Recurrent correlative auto-associative memory [18, 19]	AAM	High	Bipolar	Reconstruction of the original sample
Hopfield Neural Network [20, 21]	AAM	Low	Bipolar	Reconstruction of the original sample
Gauss machine [22, 23]	AAM	Low	Bipolar	Reconstruction of the original sample
Bidirectional associative memory [24]	AAM, HAM	Low	Bipolar	Reconstruction of the original or other sample
Brain State Model [25, 26]	AAM	-	Real	Clustering
Hamming neural network [27]	AAM	High	Bipolar	Reconstruction of the original sample
Boltzmann machine [28, 29, 30, 31]	AAM, HAM	Medium	Binary	Reconstruction of the original or other sample
ART-2 [32, 33]	AAM	-	Real	Clustering

2. MATERIALS AND METHODS

2.1. Formalize the content of the audit process of the receipt of raw materials for production and the manufactured products

Formalize the content of the supply of raw materials for production and the manufactured products are formed on the basis of audit variables (Table 2). Elements of mapping sets – data of the receipt of raw materials

for production and the manufactured products by the periods of quantization of the verification period. The vector of the receipt of raw materials features $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{iq})$ formed by indicators of quantity of raw materials by type u, $u \in U$. The vector of manufactured products features $y_i = (y_{i1}, \ldots, y_{iq})$ is formed by indicators of the quantity of manufactured products by type p, $p \in P$.

Table 2

Feature vectors for mapping raw material received - product produced

Input vector elements X		Output vector elements Y	
designation	sense	designation	sense
d	type of operation (receipt of raw materials for production)	k	type of operation (production of finished products (semi-finished products))
u	type of raw materials	p	type of product
U	set of types of raw materials	P	set of types of product
V_u	quantity of raw materials	V_p	quantity of product
$\Delta_u^{(c)}$	cost of raw material	$\Delta_{u,p}^{(c)}$	direct material costs for a product type p of a raw material type u
$\Delta^{(c)}$	total cost of raw material	$\Delta_p^{(c)}$	direct material costs for a product type p

To assess the dimension of the features vector, an analysis was made of the nomenclature of purchases of raw materials (components) of large machine-building enterprises. So, based on this analysis, we can conclude that the sections of the nomenclature are on average from 8 to 12, the number of groups in each section is from 2 to 10.

We represent the implementation of the "generalized audit" in the form of a mapping (comparison) of generalized quantitative features of the audited sets. The formation of generalized quantitative features can be performed using ANN.

2.2. Choosing a neural network model for mapping audit sets

In the work, the Full (bidirectional) Counterpropagating Neural Network (FCPNN), which is a non-recurrent static two-layer ANN, was chosen as a neural network. FCPNN output is linear.

FCPNN advantages:

1. unlike most ANNs are used to reconstruct another sample using auto-associative and hetero-associative memory.
2. unlike bidirectional associative memory and the Boltzmann machine, it works with real data.
3. unlike bidirectional associative memory and the Boltzmann machine, it has less computational complexity.

FCPNN model performing mapping of each input sample $\boldsymbol{x} = (x_1, \ldots, x_{N^x})$ to output sample $\boldsymbol{y} = (w_{i^*1}^{(2)}, \ldots, w_{i^*N^y}^{(2)})$, is represented as

$$i^* = arg\, \min_i z_i,\ z_i = \sqrt{\sum_{k=1}^{N^x} (x_k - w_{ki}^{(1)})^2},\ i \in \overline{1, N^{(1)}}, \quad (1)$$

where $w_{ki}^{(1)}$ – connection weight from the k-th element of the input sample to the i-th neuron,

$w_{i^*j}^{(2)}$ – connection weight from the neuron-winner i^* to i-th element of output sample,

$N^{(1)}$ – the number of neurons in the hidden layer.

FCPNN model performing mapping of each output sample $\boldsymbol{d} = (d_1, \ldots, d_{N^d})$ to input sample $\boldsymbol{v}_{i^*}^{(2)} = (v_{i^*1}^{(2)}, \ldots, v_{i^*N^x}^{(2)})$, is represented as

$$i^* = arg\,\min_i z_i,\ z_i = \sqrt{\sum_{s=1}^{N^d}(d_s - v_{si}^{(1)})^2},\ i \in \overline{1, N^{(1)}},$$

where $v_{si}^{(1)}$ – connection weight from the s-th element of the input sample to the i-th neuron of hidden layer,

$v_{i^*j}^{(2)}$ – connection weight from the neuron-winner i^* of hidden layer to j-th element of output sample,

$N^{(1)}$ – the number of neurons in the hidden layer.

2.3. Criterion choice for assessing the effectiveness of a neural network model for mapping audit sets

In this work for training model FCPNN was chosen target function, that indicates selection of the vector of parameter values $W = (w_{11}^{(1)}, \ldots, w_{N^xN^{(1)}}^{(1)}, w_{11}^{(2)}, \ldots, w_{N^{(1)}N^y}^{(2)})$, $V = (v_{11}^{(1)}, \ldots, v_{N^dN^{(1)}}^{(1)}, v_{11}^{(2)}, \ldots, v_{N^{(1)}N^x}^{(1)})$ which deliver the minimum mean square error (difference between the model sample and the test sample)

$$F = \frac{1}{2}\left(\frac{1}{PN^d}\sum_{\mu=1}^{P}\left\|\boldsymbol{w}_{\mu i^*}^{(2)} - \boldsymbol{d}_\mu\right\|^2 + \frac{1}{PN^x}\sum_{\mu=1}^{P}\left\|\boldsymbol{v}_{\mu i^*}^{(2)} - \boldsymbol{x}_\mu\right\|^2\right) \to \min_{W,V} \quad (2)$$

where $\boldsymbol{y}_\mu$ – μ-th, output sample according to the model,

$\boldsymbol{d}_\mu$ – μ-th test output sample,

$\boldsymbol{v}_{\mu i^*}^{(2)}$ – μ-th, input sample according to the model,

$\boldsymbol{x}_\mu$ – μ-th test input sample.

2.4. Training method for neural network model in batch mode

The disadvantage of FCPNN is that it does not have a batch learning mode, which leads to reducing of the learning speed. For FCPNN was used concurrent training (combination of training with and without a teacher). This work proposes training FCPNN in batch mode.

First phase (training of the hidden layer) (steps 1-6).

The first phase allows you to calculate the weights of the hidden layer $w_{ki}^{(1)}$ and consists of the following blocks (Fig 1).

1. Learning iteration number $n = 0$, initialization by uniform distribution on the interval (0,1) or [-0.5, 0.5] of weights $w_{ij}^{(1)}(n)$, $v_{si}^{(1)}(n)$, $i \in \overline{1, N^x}$, $j \in \overline{1, N^{(1)}}$, $s \in \overline{1, N^d}$ where N^x – is length of the sample x , N^d – is length of the sample $\boldsymbol{d}$ and $N^{(1)}$ – the number and the neurons in the hidden layer.

Training set is $\{(\boldsymbol{x}_\mu, \boldsymbol{d}_\mu) | \boldsymbol{x}_\mu \in R^{N^x}, \boldsymbol{d}_\mu \in R^{N^d}\}$, $\mu \in \overline{1, P}$, where $\boldsymbol{x}_\mu$ – μ -th training input vector, $\boldsymbol{d}_\mu$ – μ -th training output vector, P – training set power.

Initial shortest distance $\bar{z}(0)$ =0.

2. Calculating the distance to all hidden neurons.

Distance $z_{\mu i}$ from μ-th input sample to each i-th neuron and from each μ-th output sample to each i- th neuron of the hidden basis is determined by the formula:

$$z_{\mu i} = \sqrt{\sum_{k=1}^{N^x}(x_{\mu k} - w_{ki}^{(1)}(n))^2} + \sqrt{\sum_{s=1}^{N^d}(d_{\mu s} - v_{si}^{(1)}(n))^2},\ \mu \in \overline{1, P},\ i \in \overline{1, N^{(1)}}, \quad (3)$$

where $w_{ki}^{(1)}(n)$ – connection weight from k-th input sample to i-th neuron at time n,

$v_{si}^{(1)}(n)$ – pretrained connection weight from s-th element of output sample to i-th neuron of hidden layer at time n.

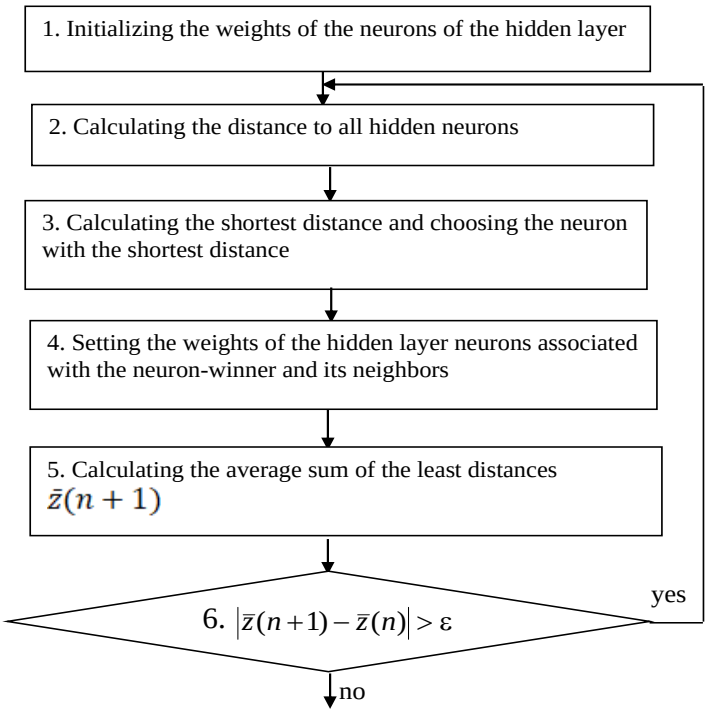


Figure 1. The sequence of steps in training method of FCPNN in batch mode (the first phase)

3. Calculating the shortest distance and choosing the neuron with the shortest distance

Calculating the shortest distance

$$z_\mu = \min_i z_{\mu i},\ \mu \in \overline{1,P},\ i \in \overline{1,N^{(1)}} \tag{4}$$

and choosing the neuron-winner i_μ^*, for which the distance $z_{\mu i}$ is shortest

$$i_\mu^* = arg\min_i z_{\mu i},\ \mu \in \overline{1,P},\ i \in \overline{1,N^{(1)}}. \tag{5}$$

4. Setting the weights of the hidden layer neurons associated with the neuron-winner i_μ^* and its neighbors based on k-means rule

$$w_{ki}^{(1)}(n+1) = \frac{\sum_{\mu=1}^{P} h(i,i_\mu^*)x_{\mu k}}{\sum_{\mu=1}^{P} h(i,i_\mu^*)},\ k \in \overline{1,N^x},\ i \in \overline{1,N^{(1)}}, \tag{6}$$

$$v_{si}^{(1)}(n+1) = \frac{\sum_{\mu=1}^{P} h(i,i_\mu^*)d_{\mu s}}{\sum_{\mu=1}^{P} h(i,i_\mu^*)},\ s \in \overline{1,N^d},\ i \in \overline{1,N^{(1)}}, \tag{6'}$$

where $h(i,i^*)$ – rectangular topological neighborhood function,

$h(i,i^*) = \begin{cases} 1, & i = i^* \\ 0, & i \neq i^* \end{cases}$.

5. Calculating the average sum of the shortest distances

$$\bar{z}(n+1) = \frac{1}{P}\sum_{\mu=1}^{P} z_\mu. \tag{7}$$

6. Checking the termination condition

If $|\bar{z}(n+1) - \bar{z}(n)| \leq \varepsilon$, the finish, else $n = n+1$, go to step 2.

Second phase (training the output layer) (steps 7-12). The second phase allows you to calculate the weights of the output layer $w_{ij}^{(2)}$ and $v_{is}^{(2)}$ and consists of the following blocks (Figure 2).

7. Learning iteration number $n = 0$, initialization by uniform distribution on the interval (0,1) or [- 0.5, 0.5] of weights $w_{ij}^{(2)}(n)$, $v_{iq}^{(2)}(n)$, $i \in \overline{1,N^{(1)}}$, $j \in \overline{1,N^d}$, $q \in \overline{1,N^x}$, where N^x – length of the input sample $\boldsymbol{x}$, N^d – length of the output sample $\boldsymbol{d}$, $N^{(1)}$ – the number and the neurons in the hidden layer.

Training set is $\{(\boldsymbol{x}_\mu, \boldsymbol{d}_\mu) | \boldsymbol{x}_\mu \in R^{N^x}, \boldsymbol{d}_\mu \in R^{N^y}\}$, $\mu \in \overline{1,P}$, where $\boldsymbol{x}_\mu$ – μ-th training input vector, $\boldsymbol{d}_\mu$ – μ -th training output vector, P – training set power.

Initial shortest distance $\bar{z}(0)$ =0.

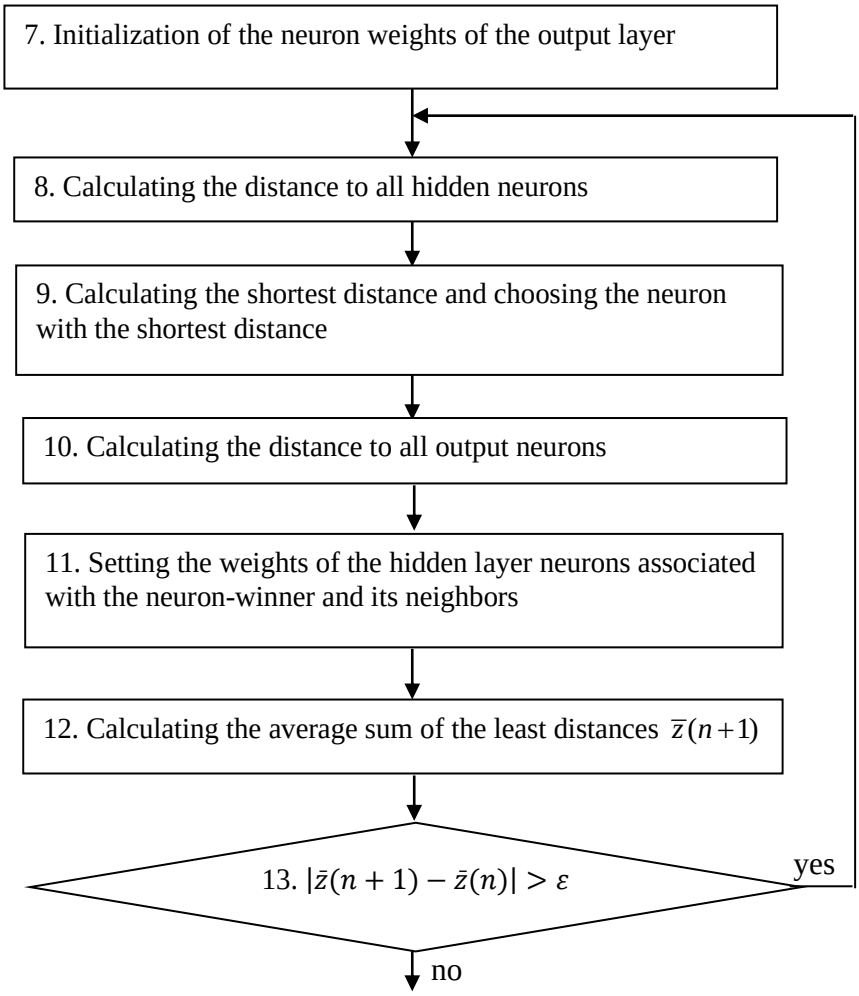


Figure 2. Sequence of procedures for the FCPNN training method in batch mode (second phase)

8. Calculating the distance to all hidden neurons

Sum of distances $z_{\mu i}$ from μ-th input sample in input layer from each *i*-th neuron of the hidden layer and from each μ-th output sample in the input layer to each *i*-th neuron of the hidden layer is determined by the formula:

$$z_{\mu i} = \sqrt{\sum_{k=1}^{N^x}(x_{\mu k} - w_{ki}^{(1)})^2} + \sqrt{\sum_{s=1}^{N^d}(d_{\mu s} - v_{si}^{(1)})^2}, \mu \in \overline{1,P}, i \in \overline{1,N^{(1)}}, \quad (8)$$

where $w_{ki}^{(1)}$ – pretrained connection weight from *k*-th element of input sample to *i*-th neuron at time n,

$v_{si}^{(1)}$ – pretrained connection weight from *s*- th element of input sample to *i*- th neuron of hidden layer at time n.

9. Calculating the shortest distance and choosing the neuron with the shortest distance.

Calculating the shortest distance

$$z_\mu = \min_i z_{\mu i}, \mu \in \overline{1,P}, i \in \overline{1,N^{(1)}} \quad (9)$$

and choosing the neuron-winner i_μ^*, for which the distance $z_{\mu i}$ is shortest.

$$i_\mu^* = arg\min_i z_{\mu i}, \mu \in \overline{1,P}, i \in \overline{1,N^{(1)}}. \quad (10)$$

10. Calculating the distance to all output neurons.

Distance z_μ from the neuron-winner i_μ^* to μ-th input and output sample in output layer is determined by the formula:

$$z_\mu = \sqrt{\sum_{j=1}^{N^d}(d_{\mu j} - w_{i^* j}^{(2)}(n))^2} + \sqrt{\sum_{r=1}^{N^x}(x_{\mu r} - v_{i^* r}^{(2)}(n))^2}, \mu \in \overline{1,P}, \quad (11)$$

where $w_{i_\mu^* j}^{(2)}(n)$ – weight of connection from the winner neuron i_μ^* of hidden layer to *j*-th element of the input sample in output layer at time n,

11. Setting the weights of the output layer neurons associated with the neuron-winner i_μ^* and its neighbors based on k-means rule

$$w_{ij}^{(2)}(n+1) = \frac{\sum_{\mu=1}^{P} h(i,i_\mu^*) d_{\mu j}}{\sum_{\mu=1}^{P} h(i,i_\mu^*)}, i \in \overline{1,N^{(1)}}, j \in \overline{1,N^y}, \quad (12)$$

$$v_{is}^{(2)}(n+1) = \frac{\sum_{\mu=1}^{P} h(i,i_\mu^*) x_{\mu i}}{\sum_{\mu=1}^{P} h(i,i_\mu^*)}, i \in \overline{1,N^{(1)}}, s \in \overline{1,N^x}, \quad (12')$$

where $h(i,i^*)$ – rectangular topological neighborhood function,

$h(i,i^*) = \begin{cases} 1, & i = i^* \\ 0, & i \neq i^* \end{cases}$.

12. Calculating the average sum of the shortest distances

$$\bar{z}(n+1) = \frac{1}{P}\sum_{\mu=1}^{P} z_\mu. \quad (13)$$

13. Checking the termination condition

If $|\bar{z}(n+1) - \bar{z}(n)| \leq \varepsilon$, the finish, else $n = n+1$, go to step 8.

2.5. Algorithm for training neuron network model in batch mode for implementation on GPU

For the proposed method of training FCPNN on audit data example, examines the algorithm for implementation on a GPU with usage of CUDA parallel processing technology.

The first phase (training the hidden layer). The first phase based on formulas (1)-(7) is shown in Fig. 3. This block diagram functions as follows.

Step 1 – Operator enters lengths N^x of the sample $\boldsymbol{x}$, the lengths N^d of the sample $\boldsymbol{d}$, the number and the neurons in the hidden layer $N^{(1)}$, power of the training set P, training set $\{(\boldsymbol{x}_\mu, \boldsymbol{d}_\mu) | \boldsymbol{x}_\mu \in R^{N^x}, \boldsymbol{d}_\mu \in R^{N^d}\}$, $\mu \in \overline{1,P}$.

Step 2 – Initialization by uniform distribution over the interval (0,1) or [-0.5, 0.5] of weights $w_{ij}^{(1)}(n)$, $v_{si}^{(1)}(n)$, $i \in \overline{1,N^x}$, $s \in \overline{1,N^d}$, $j \in \overline{1,N^{(1)}}$.

Step 3 – Calculation of distances to all hidden neurons of the ANN, using $P \cdot N^{(1)}$ threads on GPU, which are grouped into P blocks. Each thread calculates the distance from μ-th input sample to each *i*-th neuron $z_{\mu i}$.

Step 4 – Computation based on shortest distance reduction and determining the neurons with the shortest distance using $P \cdot N^{(1)}$ threads on GPU, which are grouped into P blocks. The result of the work of each block is a neuron-winner i_μ^* with the smallest distance z_μ.

Step 5 – Setting the weights of the output layer neurons associated with the neuron- winner i_μ^* and its neighbors based on reduction using $N^x \cdot N^{(1)} \cdot P$ threads on GPU, which are grouped into $N^x \cdot N^{(1)}$ blocks. The result of the work of each block is the weight $w_{ki}^{(1)}(n+1)$.

Step 6 – Setting the weights of the output layer neurons associated with the neuron- winner i_μ^* and its neighbors based on reduction using $N^d \cdot N^{(1)} \cdot P$ threads on GPU, which are grouped into $N^d \cdot N^{(1)}$ blocks. The result of the work of each block is the weight $v_{si}^{(1)}(n+1)$.

Step 7 – Calculation based on reduction of the average sum of the shortest distances using P threads on GPU, which are grouped into 1 block. The result of the block is the average sum of the smallest distances $\bar{z}(n+1)$.

Step 8 – If average sum of smallest distances of neighboring iterations are close, $|\bar{z}(n+1) - \bar{z}(n)| \leq \varepsilon$, then finish, else – increasing number of iteration $n = n+1$, go to step 3.

Step 9 – Recording of weight $w_{ki}^{(1)}(n+1)$ and $v_{si}^{(1)}(n+1)$ in the database.

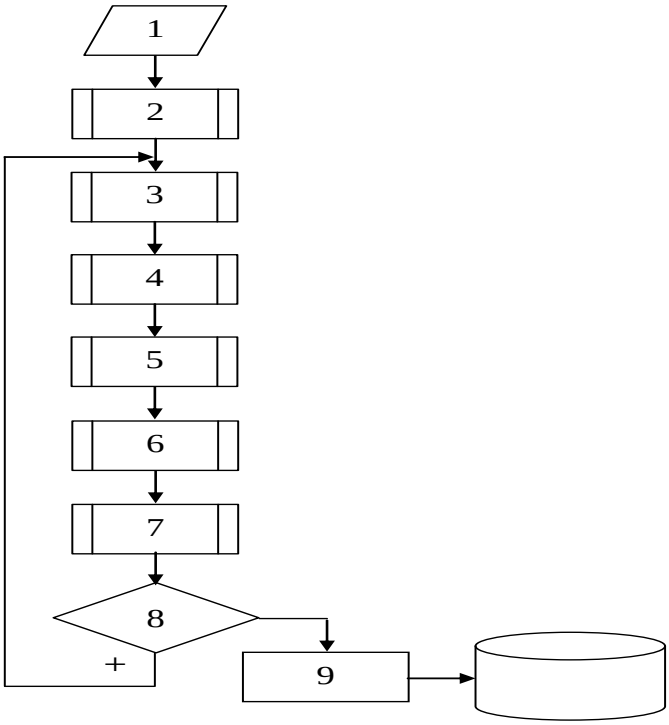


Figure 3. Block diagram of the FCPNN learning algorithm in batch mode (first phase)

Second phase (training of output layer). The second phase based on formulas (8)-(13) is showed in Figure. 4. This flowchart operates as follows.

Step 1 – Operator enters lengths N^x of the sample $\boldsymbol{x}$, the lengths N^d of the sample $\boldsymbol{d}$ the number and the neurons in the hidden layer $N^{(1)}$, power of the training set P, training set $\left\{(\boldsymbol{x}_\mu, \boldsymbol{d}_\mu)\middle|\boldsymbol{x}_\mu \in R^{N^x}, \boldsymbol{d}_\mu \in R^{N^d}\right\}, \mu \in \overline{1,P}$.

Step 2 – Initialization by uniform distribution over the interval (0,1) or [-0.5, 0.5] of weights $w_{ij}^{(2)}(n)$, $v_{iq}^{(2)}(n)$, $i \in \overline{1,N^{(1)}}$, $j \in \overline{1,N^d}$, $q \in \overline{1,N^x}$.

Step 3 – Calculation of distances to all hidden neurons of the ANN, using $P \cdot N^{(1)}$ threads on GPU, which are grouped into P blocks. Each thread calculates the distance from μ-th input sample to each i-th neuron $z_{\mu i}$.

Step 4 – Computation based on shortest distance reduction and determining the neurons with the shortest distance using $P \cdot N^{(1)}$ threads on GPU, which are grouped into P blocks. The result of the work of each block is a neuron-winner i_μ^* with the smallest distance z_μ.

Step 5 – Calculating distances from the neuron-winner i_μ^* to μ-th output sample using $P \cdot N^{(d)}$ threads on GPU, which are grouped into P blocks. Each thread calculates the distance from the neuron-winner i_μ^* to μ-th output sample z_μ.

Step 6 – Setting the weights of the output layer neurons associated with the neuron-winner i_μ^* and its neighbors based on reduction using $N^{(1)} \cdot N^d \cdot P$ threads on GPU, which are grouped into $N^{(1)} \cdot N^d$ blocks. The result of the work of each block is the weight $w_{ij}^{(2)}(n+1)$.

Step 7 – Setting the weights of the output layer neurons associated with the neuron-winner i_μ^* and its neighbors based on reduction using $N^{(1)} \cdot N^x \cdot P$ threads on GPU, which are grouped into $N^{(1)} \cdot N^x$ block. The result of the block is the average sum of the smallest distances $v_{is}^{(2)}(n+1)$.

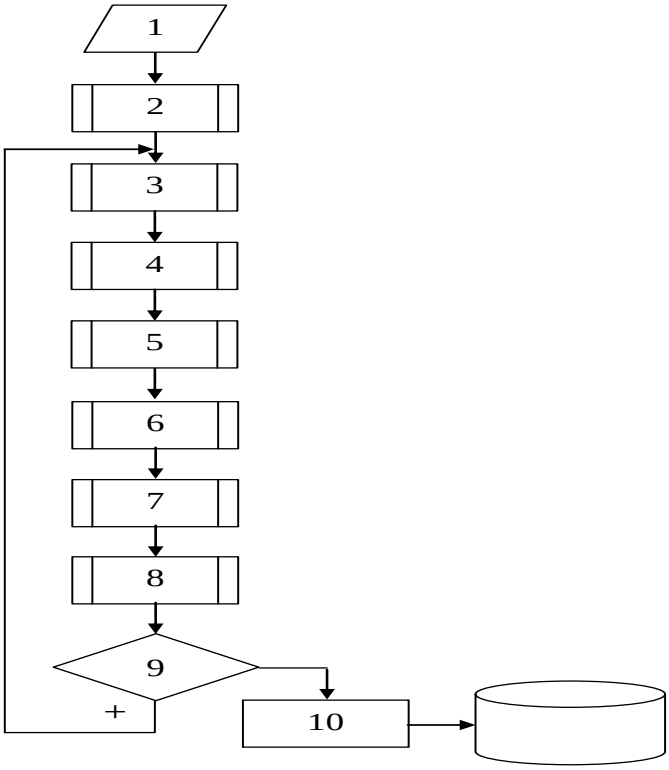


Figure 4. Block diagram of the FCPNN training algorithm in batch mode (second phase)

Step 8 – Calculation based on reduction of the average sum of the shortest distances using P threads on GPU, which are grouped into 1 block. The result of the block is the average sum of the smallest distances $\bar{z}(n+1)$.

Step 9 – If average sum of smallest distances of neighboring iterations are close, $|\bar{z}(n+1) - \bar{z}(n)| \leq \varepsilon$, then finish, else – increasing number of iteration $n = n + 1$, go to step 3.

Step 10 – Recording of weight $w_{ij}^{(2)}(n+1)$ and $v_{is}^{(2)}(n+1)$, in the database.

2.6. Numerical research

The results of the comparison of the proposed method using GPU and the traditional FCPNN training method are presented in Table 3.

Table 3

Comparison of the computational complexity of the proposed and traditional training methods of FCPNN

Feature	Method	
	proposed	traditional
Computational complexity	$O(n_1^{max} + n_2^{max})$	$O(PN^{(1)}n_1^{max} + (PN^{(1)} + P)n_2^{max})$

Evaluation of computational complexity of the proposed method using the GPU, and the traditional method of teaching FOCPNN were based on the number of calculation distances, computing of which is the most consuming part of method. Moreover, n_1^{max} – the maximum number of iterations of the first training phase, n_2^{max}– the maximum number of iterations of the second training phase, $N^{(1)}$ – the number of neurons in the hidden layer, P – the power of the training set.

2.7. Discussion

The traditional FCPNN learning method does not provide support for batch mode, which increases computational complexity (Table 3). Proposed method eliminates this flaw and allows for approximate increase of learning rate in $PN^{(1)}$.

2.8. Conclusion

1. The urgent task of increasing the effectiveness of audit in the context of large volumes of analyzed data and limited verification time was solved by automating the formation of generalized features of audit sets and their mapping by means of a full bidirectional counterpropagating neural network.

2. For increased learning rate of full bidirectional counterpropagating neural network, was developed a method based on the k-means rule for training in batch mode. The proposed method provides: approximately increase learning rate in $PN^{(1)}$, where $N^{(1)}$ is the number of neurons in the hidden layer and P is the power of the learning set.

3. Created a learning algorithm based on k-means, intended for implementation on a GPU using CUDA technology.

4. The proposed method of training based on the k-means rule can be used to intellectualize the DSS audit.

Prospects for further research is the study of the proposed method for a wide class of artificial intelligence tasks, as well as the creation of a method for bidirectional mapping audit features to solve audit problems.

3. References

[1] The World Bank: World Development Report 2016: Digital Dividends. https://www.worldbank.org/en/publication/wdr2016 (2016). Accessed 12 February 2020

[2] T.V. Neskorodieva. Postanovka elementarnykh zadach audytu peredumovy polozhen bukhhalterskoho obliku v informatsiinii tekhnolohii systemy pidtrymky rishen (Formulation of elementary tasks of audit subsystems of accounting provisions precondition IT DSS). Modern information systems. 3(1), 48-54, 2019. doi:10.20998/2522-9052.2019.1.08

[3] T.V. Neskorodieva. Formalization method of the first level variables in the audit systems IT ISSN 1028-9763. Matematychni mashyny i systemy (Mathematical machines and systems), № 4, 2019. doi: 10.34121/1028-9763-2019-4-79-86

[4] R. Heht-Nielsen. Counterpropagating networks. Proc. Int. Conf. on Neural Networks, New York, NY, volume 2, pp. 19-32, 1987.

[5] R. Heht-Nielsen. Application counterpropagating networks. Neural Networks, vol. 1, pp. 131-139, 1988.

[6] Sivanandam S.N., Sumathi S., Deepa S.N. Introduction to Neural Networks using Matlab 6.0. New Delhi: The McGraw-Hill Comp., Inc., 2006.

[7] R.M. Neal Connectionist learning of belief networks. Artificial Intelligence, vol. 56, pp. 71-113, 1992.

[8] P. Dayan, B.J. Frey, R.M. Neal. The Helmholtz machine. Neural Networks, vol. 7, pp. 889-904, 1995.

[9] G.E. Hinton. The ‘wake-sleep’ algorithm for unsupervised neural networks. Science, vol. 268, pp. 1158-1161, 1995.

[10] T. Kohonen. Self-organizing Maps. Berlin: Springer-Verlag, 1995.

[11] M. Martinez, S. Berkovich, K. Schulten. «Neural-gas» network for vector quantization and its application to time series prediction. IEEE Trans. on Neural Networks, vol. 4, pp. 558-569, 1993.

[12] S. Haykin. Neural Networks and Learning Machines. Upper Saddle River, NJ: Pearson Education, Inc., 2009.

[13] T.D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural Networks, vol. 2, pp. 459-473, 1989.

[14] M.S. Bartlett, J.R. Movellan, T.J. Sejnowski. Face Recognition by Independent Component Analysis. IEEE Trans. on Neural Networks, vol. 13, № 6. pp. 1450-1464, 2002.
[15] B. Draper, K. Baek, M.S. Bartlett, J.R. Beveridge. Recognizing Faces with PCA and ICA. Computer Vision and Image Understanding (Special Issue on Face Recognition), vol. 91, Issues 1-2, pp. 115-137, 2003.
[16] J.S. Albus A new approach to manipulator control: the cerebellar model articulation controller (CMAC). Journal of dynamic systems, measurement, and control trans. ASME, vol. 97, № 6. pp. 228-233, 1975.
[17] T.D. Chiueh, R.M. Goodman Recurrent correlation associative memories. IEEE Transactions on Neural Networks, vol. 2, № 2. pp. 275–284, 1991.
[18] T.D. Chiueh, R.M. Goodman. High capacity exponential associative memories. Proc. IEEE Int. Conf. Neural Networks. San Diego. CA., vol. 1. pp. 153-160, 1988.
[19] J.J. Hopfield Neural networks and physical systems with emergent collective computation abilities. Proc. Nac. Academy of Sciences USA, vol.79. pp. 2554-2558, 1982.
[20] J.J. Hopfield, Tank D.W. Neural computation of decisions in optimization problems. Biolog. Cybern, vol.52. pp. 141-152, 1985.
[21] Y. Akiyama, A. Yamashita, M. Kajiura, H. Aiso. Combinational optimization with Gaussian machines. Neural Networks. vol. 1. pp. 533-540, 1989.
[22] P.S. Neelakanta, D. DeGroff. Neural network modelling: statistical mechanics and cybernetic perspectives Boca Raton, Florida: CRC Press, 1994.
[23] B. Kosko Bidirectional associative memories. IEEE Trans. on Syst., Man and Cybern, vol.18. pp.49-60, 1988.
[24] J.A. Anderson, J.W. Silverstein, S.A. Ritz, R.S. Jones. Distinctive features, categorical perception and probability learning: Some applications of a neural models. Psychological Review, vol.84. pp.413-451, 1977.
[25] J.A. Anderson. Cognitive and psychological computation with neural models. IEEE Trans. on Syst., Man and Cybern, vol.13. pp.799-815, 1983.
[26] R.P. Lippmann An introduction to computing with neural nets. IEEE Acoustics, Speech and Signal Processing Magazine, April. pp. 4-22, 1987.
[27] A. Fischer, C. Igel. Training Restricted Boltzmann Machines: An Introduction Pattern Recognition, vol. 47. pp.25-39, 2014.
[28] N. Srivastava, R.R. Salakhutdinov. Multimodal Learning with Deep Boltzmann Machines. Journal of Machine Learning Research, vol. 15. pp. 2949-2980, 2014.
[29] R.R. Salakhutdinov, G.E. Hinton. Deep Boltzmann machines. Journal of Machine Learning Research, vol. 5. pp. 448-455, 2009.
[30] R.R. Salakhutdinov, H. Larochelle. Efficient Learning of Deep Boltzmann Machines. Journal of Machine Learning Research, vol. 9, pp. 693-700, 2010.
[31] G.A. Carpenter, S. Grossberg. ART-2: self-organization of stable category recognition codes for analog input patterns. Applied Optics, vol.26, pp.4919-4930, 1987.
[32] G.A. Carpenter, S. Grossberg. ART-3: self-organization of stable category recognition codes for analog input patterns. Neural, vol.3, pp. 129-152, 1990.