Automatic Analysis Method of Audit Data Based on 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

The work solves the problem of increasing the efficiency and effectiveness of analytical audit procedures by automating data comparison by neural network mapping. The object of the research is the process of auditing the compliance of payment sequences and supply for raw materials. The vectors of signs for the objects of the sequences of payment and supply of raw materials 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 forward-only 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.

Keywords

audit, mapping by neural network, forward-only counterpropagating neural network, sequences of payment and supply of raw materials.

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.

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.

IT&I-2020 Information Technology and Interactions, December 02–03, 2020, KNU Taras Shevchenko, Kyiv, Ukraine
EMAIL: t.neskorodieva@donnu.edu.ua (T. Neskorodieva); fedorovee75@ukr.net (E. Fedorov)
ORCID: 0000-0003-2474-7697 (T. Neskorodieva); 0000-0003-3841-7373 (E. Fedorov)

2. Problem Statement

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 aim of the work is to increase the efficiency of automatic data analysis in the audit DSS by means of a neural network mapping of sets of audit indicators in order to identify systematic misstatements that lead to misstatement of reporting. It is assumed that the audit indicators are noisy with Gaussian noise, which in turn simulates random accounting errors (as opposed to systematic ones).

For the achievement of the aim it is necessary to solve the following tasks:

- generate vectors of indicators for objects of sequences of payment and supply of raw materials;
- 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.

3. Literature Survey

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. cannot be used for reconstruction of the other sample;
2. do not work with real data.

4. Materials and Methods

4.1. Formation of features vectors of elements of payment sequences and raw materials supply

Feature of elements of the sequence of payment and supply of raw materials are formed on the basis of audit variables (Table 2). Elements of mapping sets - payment (delivery) data for each supplier with which a long-term supply agreement is in force during the year for which the audit is carried out. The vector of payment signs $\boldsymbol{x}_i = (x_{i1},\ldots,x_{iq})$ formed by indicators of the cost of paid raw materials δ_{s_d} by type $s_d \in \Theta_d$. The vector of supply features $y_i = (y_{i1},\ldots,y_{iq})$ is formed by indicators of the quantity of supplied raw materials by type. v_{s_k} by type $s_k \in \Theta_k$.

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

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 2-10.

Analyzing the homogeneity of the procurement nomenclature, we can conclude that for continuous operation, a plant can have long-term contracts with suppliers in quantities from 50 to 100.

Table 2
Feature vectors for mapping paid-received

Input vector elements X		Output vector elements Y	
designation	sense	designation	sense
d	type of operation (payment of supplier invoice)	k	type of operation (receipt of raw materials from a supplier)
p_d	type of supplier to whom payment is transferred	p_k	type of supplier who supplied raw materials
s_d	type of paid raw materials	s_k	type of raw material received
θ_d	set of types of paid raw materials	θ_k	set of types of raw materials obtained
q_d	number of types of paid raw materials	q_k	number of types of raw materials obtained
v_{s_d}	number of paid raw materials s_d $(s_d \in \Theta_d)$	v_{s_k}	number of received raw material s_k $(s_k \in \Theta_k)$
c_{s_d}	price of paid raw material s_d $(s_d \in \Theta_d)$	c_{s_k}	price of the received raw material s_k $(s_k \in \Theta_k)$
δ_{s_d}	price of paid raw material s_d $(s_d \in \Theta_d)$	δ_{s_k}	price of the received raw material s_k $(s_k \in \Theta_k)$
δ_d	cost of paid raw material set Θ_d	δ_k	cost of the received raw material set Θ_k
t_d	maximum delivery time according to the contract	t_k	delivery lag after the fact

Based on the analysis performed, the following quantitative features were selected for the elements of the supply sequence for each supplier and for all types of raw materials (the order of quantity of the assortment of raw materials is described above):

x_j - cost of paid raw material by type j $(j = \overline{1,q})$.

For the elements of the supply chain for each supplier and for all types of raw materials, the following features have been selected:

x_j - amount of received raw material by type j $(j = \overline{1,q})$.

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.

4.2. Choosing a neural network model for mapping audit sets

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

FOCPNN advantages:

1. Unlike most ANNs are used to reconstruct another sample using hetero-associative memory.
2. Unlike bidirectional associative memory and the Boltzmann machine, it works with real data.
3. Unlike a full counterpropagating neural network, it has less computational complexity (it does not perform additional reconstruction of the original sample).

FOCPNN 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 j-th element of output sample,
$N^{(1)}$ – the number of neurons in the hidden layer.

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

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

$$F = \frac{1}{PN^y}\sum_{\mu=1}^{P}\left\|\boldsymbol{y}_\mu - \boldsymbol{d}_\mu\right\|^2 \to \min_W, \tag{2}$$

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

4.4. Training method for neural network model in batch mode

The disadvantage of FOCPNN is that it does not have a batch learning mode, which leads to reducing of the learning speed. For FOCPNN was used concurrent training (combination of training with and without a teacher). This work proposes training FOCPNN 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 $w_{ij}^{(1)}(n)$, $i \in \overline{1,N^x}$, $j \in \overline{1,N^{(1)}}$, where N^x – is length of the sample x and $N^{(1)}$ – the number and the neurons in the hidden layer.

Training set is $\{\boldsymbol{x}_\mu | \boldsymbol{x}_\mu \in R^{N^x}\}$, $\mu \in \overline{1,P}$, where $\boldsymbol{x}_\mu$ – μ -th training input 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 is determined by the formula:

$$z_{\mu i} = \sqrt{\sum_{k=1}^{N^x}(x_{\mu k} - w_{ki}^{(1)}(n))^2},\ \mu \in \overline{1,P},\ i \in \overline{1,N^{(1)}}, \tag{3}$$

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

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

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

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

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

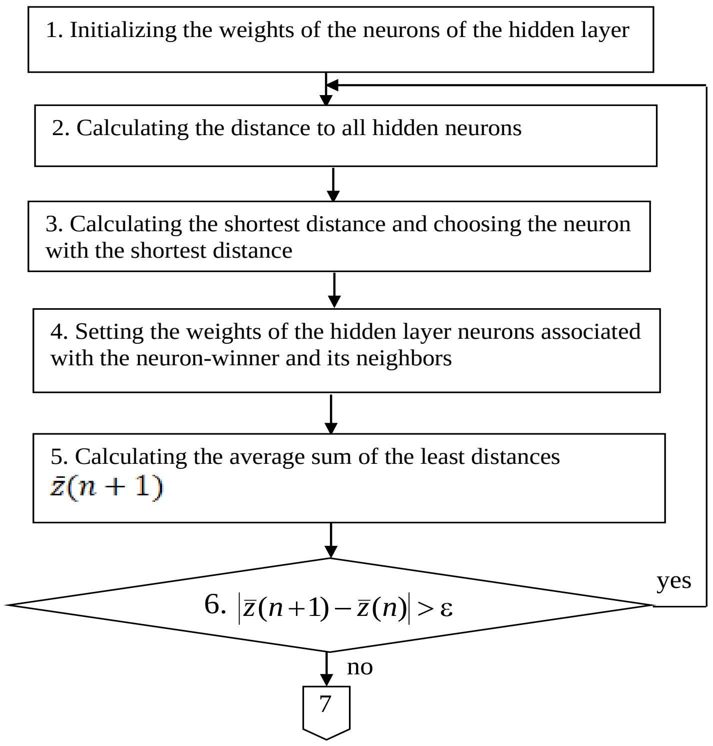
1. Initializing the weights of the neurons of the hidden layer
2. Calculating the distance to all hidden neurons
3. Calculating the shortest distance and choosing the neuron with the shortest distance
4. Setting the weights of the hidden layer neurons associated with the neuron-winner and its neighbors
5. Calculating the average sum of the least distances $\bar{z}(n+1)$
6. $|\bar{z}(n+1)-\bar{z}(n)|>\varepsilon$
yes
no
7

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

6. Checking the termination condition

If $|\bar{z}(n+1)-\bar{z}(n)| \le \varepsilon$, 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 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 $w_{ij}^{(2)}(n)$, $i \in \overline{1,N^{(1)}}$, $j \in \overline{1,N^{y}}$, where $N^{(1)}$ – the number and the neurons in the hidden layer, N^{y} – length of the sample $\boldsymbol{m}_y$.

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, N^x – length of the sample $\boldsymbol{x}$.

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

8. Calculating the distance to all hidden neurons

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

$$z_{\mu i} = \sqrt{\sum_{k=1}^{N^x}(x_{\mu k} - w_{ki}^{(1)})^2},\ \mu \in \overline{1,P},\ i \in \overline{1,N^{(1)}}, \tag{8}$$

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

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

Calculating the shortest distance

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

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

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

10. Calculating the distance to all output neurons.

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

$$z_\mu = \sqrt{\sum_{j=1}^{N^y}(d_{\mu j} - w_{i_\mu^* j}^{(2)}(n))^2},\ \mu \in \overline{1,P}, \tag{11}$$

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

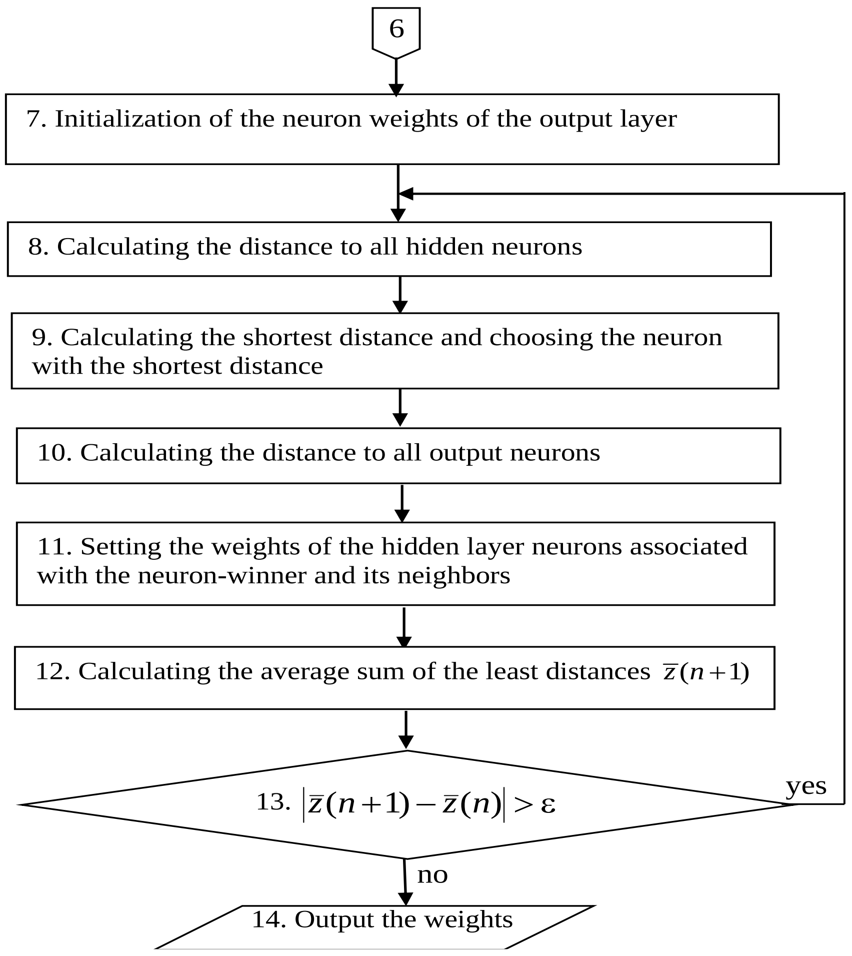
6
7. Initialization of the neuron weights of the output layer
8. Calculating the distance to all hidden neurons
9. Calculating the shortest distance and choosing the neuron with the shortest distance
10. Calculating the distance to all output neurons
11. Setting the weights of the hidden layer neurons associated with the neuron-winner and its neighbors
12. Calculating the average sum of the least distances $\bar{z}(n+1)$
13. $|\bar{z}(n+1)-\bar{z}(n)|>\varepsilon$
yes
no
14. Output the weights

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

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

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

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

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

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.

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

For the proposed method of training FOCPNN 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 s of the sample $\boldsymbol{x}$ N^x, the lengths s of the sample $\boldsymbol{y}$ N^y, the number and the neurons in the hidden layer $N^{(1)}$, power of the training set P, training set $\{\boldsymbol{x}_\mu|\boldsymbol{x}_\mu\in$

$R^{N^x}\}$, $\mu \in \overline{1,P}$.

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

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 $z_{\mu i}$.

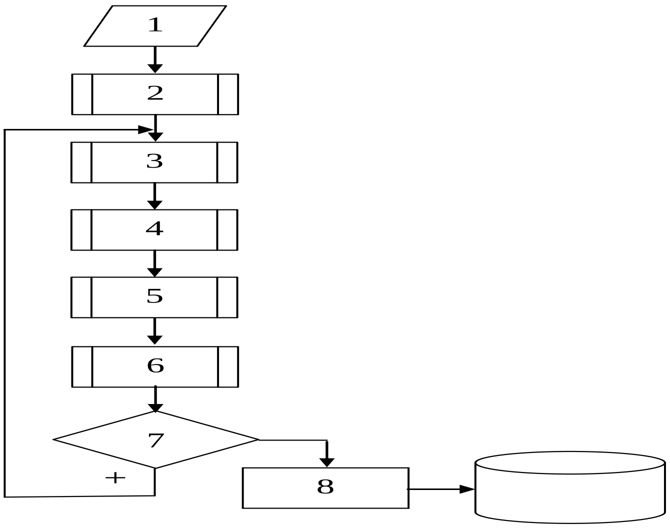

Figure 3. Block diagram of the FOCPNN learning algorithm in batch mode (first phase)

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 – 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 7 – 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 8 – Recording of weight $w_{ki}^{(1)}(n+1)$ in the database.

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

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

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.

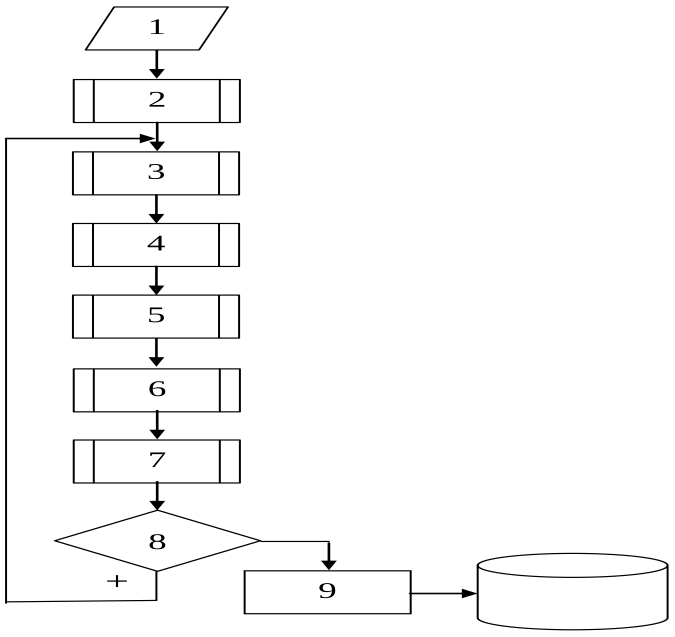

Figure 4. Block diagram of the FOCPNN training algorithm in batch mode (second phase)

Step 5 – Calculating distances from the neuron-winner i^*_μ to μ-th output sample using $P \cdot N^y$ 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 $N^{(1)} \cdot N^y \cdot P$ threads on GPU, which are grouped into $N^{(1)} \cdot N^y$ blocks. The result of the work of each block is the weight $w^{(2)}_{ij}(n+1)$.

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, $|\bar{z}(n+1) - \bar{z}(n)| \le \varepsilon$, then finish, else – increasing number of iteration $n = n + 1$, go to step 3.

Step 9 – Recording of weight $w^{(2)}_{ij}(n+1)$, in the database.

4.6. Numerical research

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

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.

Comparison of computational complexity for $P = 1000$ $n_1^{\max} = 100$, $n_2^{\max} = 100$, $N^{(1)} = 100$ showed the result of reducing the computation time in comparison with the traditional method by a factor of 100 000.

The sampling rate is inversely proportional to the quantization step of the audit data and depends on the audit period and level. Increasing the sampling rate of data acquisition leads to a directly proportional increase in the power of the training set. which increases the computational complexity by the same proportionality coefficient in the sequential learning mode, and in the parallel mode the speed does not decrease significantly.

Table 3
Comparison of the computational complexity of the proposed and traditional training methods of FOCPNN

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

4.7. Discussion

The traditional FOCPNN 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 $PN^{(1)}$.

4.8. 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 forward-only counterpropagating neural network.
2. For increased learning rate of forward-only 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 mapping audit features to solve audit problems.

5. References

[1] The World Bank: World Development Report 2016: Digital Dividends. 2016. URL: https://www.worldbank.org/en/publication/wdr2016

[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 (In Russian).

[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, Vol. 2, (1987): 19-32.

[5] R. Heht-Nielsen. Application counterpropagating networks. Neural Networks, Vol. 1, (1988): 19-32. DOI.10.1016/0893-6080(88)90015-9

[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. DOI.10.1007/978-3-540-35781-0

[7] R.M. Neal Connectionist learning of belief networks. Artificial Intelligence, Vol. 56, (1992): 71-113. DOI.10.1016/0004-3702(92)90065-6

[8] P. Dayan, B.J. Frey, R.M. Neal. The Helmholtz machine. Neural Networks, Vol. 7, (1995): 889-904. DOI.10.1162/neco.1995.7.5.889

[9] G.E. Hinton. The ‘wake-sleep’ algorithm for unsupervised neural networks. Science, Vol. 268, (1995): 1158-1161. DOI.10.1126/science.7761831
[10] T. Kohonen. Self-organizing Maps. Berlin: Springer-Verlag, 1995. DOI.10.1007/978-3-642-97610-0
[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, July 1993, Vol. 4, (1993): 558-569. DOI.10.1109/72.238311
[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, (1989): 459-473. DOI.10.1016/0893-6080(89)90044-0
[14] M.S. Bartlett, J.R. Movellan, T.J. Sejnowski. Face Recognition by Independent Component Analysis. IEEE Trans. on Neural Networks, Vol. 13, Issue 6, (2002): 1450-1464. DOI.10.1109/tnn.2002.804287
[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, (2003): 115-137. DOI.10.1016/s1077-3142(03)00077-8
[16] J.S. Albus A new approach to manipulator control: the cerebellar model articulation controller. Journal of dynamic systems, measurement, and control trans. ASME, Vol. 97, Issue 6, (1975): 228-233. DOI.10.1115/1.3426922
[17] T.D. Chiueh, R.M. Goodman Recurrent correlation associative memories. Transactions on Neural Networks, IEEE, March 1991, Vol. 2, Issue 2. (1991): 275–284. DOI.10.1109/72.80338
[18] T.D. Chiueh, R.M. Goodman. High capacity exponential associative memories. Int. Conf. Neural Networks. IEEE, 24-27 July 1988, San Diego. CA., Vol. 1, (1988): 153-160. DOI.10.1109/icnn.1988.23843
[19] J.J. Hopfield Neural networks and physical systems with emergent collective computation abilities. Proc. Nac. Academy of Sciences USA, Vol.79, (1982): 2554-2558. DOI.10.1073/pnas.79.8.2554
[20] J.J. Hopfield, Tank D.W. Neural computation of decisions in optimization problems. Biolog. Cybern, Vol.52, (1985): 141-152.
[21] Y. Akiyama, A. Yamashita, M. Kajiura, H. Aiso. Combinational optimization with Gaussian machines. Neural Networks. International 1989 Joint Conference on Neural Networks, IEEE, Washington, DC, USA, USA Vol. 1, (1989), 533-540. DOI.10.1109/ijcnn.1989.118630
[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, (1988), 49-60. DOI.10.1109/21.87054
[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, (1977), 413-451. DOI.10.1037/0033-295x.84.5.413
[25] J.A. Anderson. Cognitive and psychological computation with neural models. IEEE Trans. on Syst., Man and Cybern, Vol.13, (1983): 799-815. DOI.10.1109/tsmc.1983.6313074
[26] R.P. Lippmann An introduction to computing with neural nets. IEEE Acoustics, Speech and Signal Processing Magazine, April. (1987): 4-22. DOI.10.1109/massp.1987.1165576
[27] A. Fischer, C. Igel. Training Restricted Boltzmann Machines: An Introduction Pattern Recognition, Vol. 47, (2014): 25-39. DOI.10.1016/j.patcog.2013.05.025
[28] N. Srivastava, R.R. Salakhutdinov. Multimodal Learning with Deep Boltzmann Machines. Journal of Machine Learning Research, Vol. 15, (2014): 2949-2980.
[29] R.R. Salakhutdinov, G.E. Hinton. Deep Boltzmann machines. Journal of Machine Learning Research, Vol. 5, (2009): 448-455.
[30] R.R. Salakhutdinov, H. Larochelle. Efficient Learning of Deep Boltzmann Machines. Journal of Machine Learning Research, Vol. 9, (2010): 693-700.
[31] G.A. Carpenter, S. Grossberg. ART-2: self-organization of stable category recognition codes for analog input patterns. Applied Optics, Vol.26, (1987): 4919-4930. DOI:10.1364/ao.26.004919
[32] G.A. Carpenter, S. Grossberg. ART-3: self-organization of stable category recognition codes for analog input patterns. Neural, Vol.3, (1990): 129-152. DOI:10.1016/0893-6080(90)90085-y