EasyChair Preprint

№ 451

The Hierarchical Learning Algorithm for Deep Neural Netwprks

Stanisław Płaczek and Aleksander Płaczek

August 24, 2018

The Hierarchical Learning Algorithm for Deep Neural Networks

Stanislaw Placzek[1] and Aleksander Placzek[2]

[1] University of Business in Wroclaw, Poland
stanislaw.placzek@wp.pl
[2] Silesian University of Technology, Faculty of Automatic Control, Electronic and Computer Science, WASKO Gliwice, Poland
a.placzek@wasko.pl

Abstract

In the article, the emphasis is put on the modern artificial neural network (ANN) structure, which in the literature is known as a deep neural network. A network includes more than one hidden layer and comprises many standard modules with ReLu nonlinear activation function. A learning algorithm includes two standard steps, forward and backward, and its effectiveness depends on the way the learning error is transported back through all the layers to the first layer. Taking into account all the dimensionalities of matrixes and the nonlinear characteristics of ReLu activation function, the problem is very challenging. In practice tasks, a neural networks internal layer matrixes with ReLu activations function, include a lot of null value of weight coefficients. This phenomenon has a negative impact on the effectiveness of the learning algorithm's convergence. Analyzing and describing an ANN structure, one usually finds that the first parameter is the number of ANNs layers "L". By implementing the hierarchical structure to the learning algorithm, an ANN structure is divided into sub-networks. Every sub-network is responsible for finding the optimal value of its weight coefficients using a local target function to minimize the learning error. The second coordination level of the learning algorithm is responsible for coordinating the local solutions and finding the minimum of the global target function. In each iteration the coordinator has to send coordination parameters into the first level of subnetworks. By using the input and the teaching vectors, the local procedures are working and finding their weight coefficients. At the same step the feedback error is calculated and sent to the coordinator. The process is being repeated until the minimum of all the target functions is achieved.

1 Deep neural network structure

A deep neural network is built with topologically and logically uniform modules known as layers. A networks structure includes an input layer, several hidden layers, and finally an output layer. Usually, a network is built with $1 \div L$ layers of different or identical structure. From a mathematical point of view, a layer could be described by a matrix of weight coefficients W^l, an input vector $X^{(l-1)}$, a hidden vector U^l, an activation function $ReLu(U) = max(0, U)$, and an output vector X^l (Fig.1).

A deep neural network includes $L \div 1$ hidden layers and one output layer which contains a different activation function. An output layer needs to aggregate a set of partial features from previous layers to achieve the final result, that is an output signal. As presented in Fig.1, one can define the target function:

$$\Phi = \frac{1}{2} \cdot (X^L - Y)^T \cdot (X^L - Y) \tag{1}$$

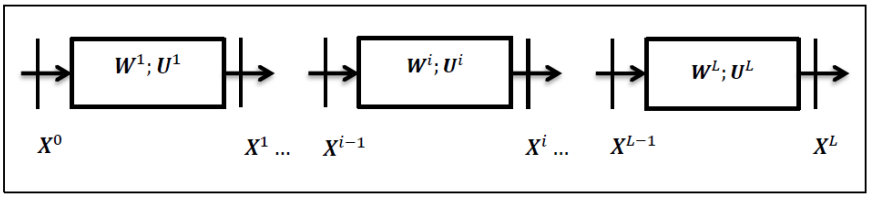

Figure 1: Deep Neural Network structure

Where:
$X^L[1 \div N^L]$-the output vector,
N^L-the dimensionality of the output vector,
$Y[1 : N^L]$-the vector of teaching data.

For all the hidden layers and for forward calculation, one can write:

$$U^l = W^l \cdot X^{l-1} \tag{2}$$

$$X^l = F(U^l) \tag{3}$$

Where:
$l = 1 \div L$- number of layers in a deep neural network,
U^l- the internal vector for layer l,
F - the vector of activation function,
W^l-the matrix of weight coefficients for layer l,
X^{l-1}, X^l - the input and output vector of layer l, accordingly.

The process of selecting an activation function is an essential and difficult task. When building standard networks, usually sigmoid and tanh activation functions are used. A sigmoid function has two areas of value in which the function, in an asymptotic way, achieves the value of zero or one. This characteristic has a negative impact on the derivative value and, at the same time, on the algorithm convergence. At the moment, a new activation function is used in a deep neural network, a Rectified Linear Unit: $ReLu$, which is defined as follows:

$$f = ReLu(u) = max(0, u) \tag{4}$$

From a mathematical point of view, its gradient is discontinuous for $u = 0$. In a computers application, this problem is solved by the accepted value $f = 0$ for $u = 0$.

1.1 Learning algorithm

In computer applications,the back propagation learning algorithm is the most popular one. From (1), one can calculate the first derivatives for the output layer, denoting:

$$\frac{\partial \Phi}{\partial X^L} = E^L = X^L - Y \tag{5}$$

$$\frac{\partial \Phi}{\partial W^L} = \frac{\partial \Phi}{\partial X^L} \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial W^L} = E^L \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial W^L} \tag{6}$$

Where:
$E^L = [\epsilon_1^L, \epsilon_2^L, ...\epsilon_{N^L}^L]$-the vector of an output layer error.

The derivatives of the ReLu function could be written as follows:

$$\frac{\partial X^L}{\partial U^L} = max(0, U^L)' = 1(U^L) \tag{7}$$

The last part of equation (6) is calculated:

$$\frac{\partial U^L}{\partial W^L} = X^{L-1} \tag{8}$$

Finally, formula (6) can be written in the matrix form using the Hadamard $\odot$ product notation :

$$\frac{\partial \Phi}{\partial W^L} = \{E^L \odot 1(U)^L\} \cdot (X^{L-1})^T \tag{9}$$

$$\frac{\partial \Phi}{\partial X^{L-1}} = \frac{\partial \Phi}{\partial X^L} \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial X^{L-1}} = E^L \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial X^{L-1}} \tag{10}$$

Using the same notation for an output layer, formula (10) can be written as:

$$E^{L-1} = \frac{\partial \Phi}{\partial X^{L-1}} = E^L \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial X^{L-1}} \tag{11}$$

In the matrix form:

$$E^{L-1} = (W^L)^T \cdot \{1(U)^L \odot E^L\} \tag{12}$$

The output layer error E^L has to be translated into the previous layer $L-1$, this process will be repeated up to the first hidden layer. Fig. 3. shows the full scheme for a back propagation algorithm and how a layer's error is translated back through the network. The final back propagation formulas have a recurrent structure. An algorithm will start from the output layer L and going through all the hidden layers, will achieve the first hidden layer 1:

$$E^{l-1} = (W^l)^T \cdot \{1(U)^l \odot E^l\} \tag{13}$$

$$\frac{\partial \Phi}{\partial W^l} = \{E^l \odot 1(U)^l\} \cdot (X^{l-1})^T \tag{14}$$

Where:
$l = 1 : L$.
According to Fig. 2. the layer is laid between input vector $X^{(l-1)}$ and output vector X^l. The same border is used for the back propagation error from E^l to $E^{(l-1)}$. In a standard neural network, a sigmoid activation function is used. An output error is translated back to the first layer decreasing its value and finally, an algorithm can calculate zero's value. This property has a very negative impact on convergence. Therefor, it is the main reason to use $ReLu$ activation function, especially in a deep neural network. Its derivative is equal to 1 or to 0 - the Heaviside step function. Some derivatives (14) are equal to 0 and the weight coefficient does not change in the actual iteration process (formula 14). The same can be observed in the error back propagation function (formula 13).

Taking into account all the limitations mentioned above, a neural network learning algorithm is decomposed, and a new coordination level is implemented. Two or more levels could be used to improve the learning algorithm convergence.

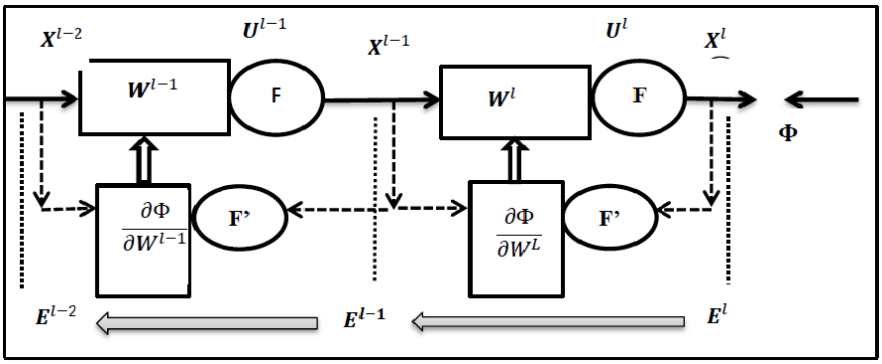

Figure 2: A scheme of an error back propagation through the layers

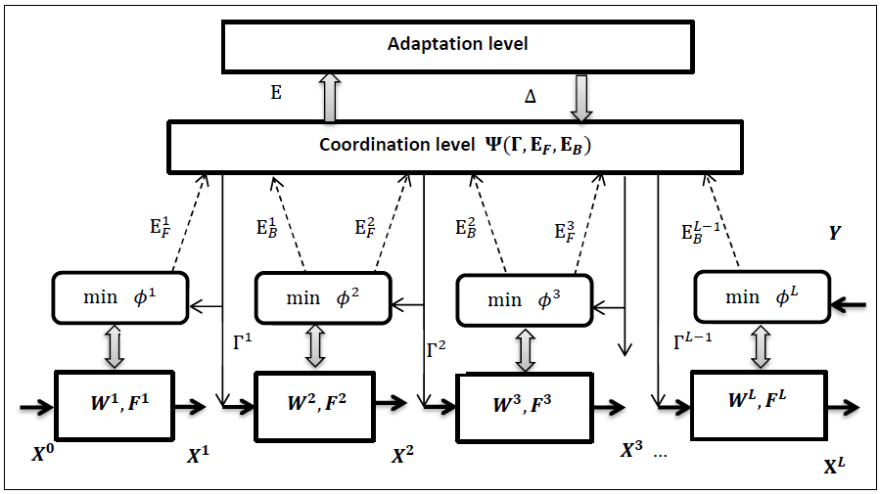

Figure 3: A scheme of an error back propagation through the layers

2 Learning algorithm decomposition

For a multi-layer ANN, a lot of hidden layers and one output layer are individual entities. The smaller part will be described as a sub-network. Every sub-network has its own output vector, an input vector of the succeeding one X^l, and a local target function Φ^l where $l = 1 \div L$. Because of the specific organization of an ANNs hierarchy there are many sub-networks on the first level, for each of which local target functions are defined:

$$\Phi = \{\Phi^1, \Phi^2 .... \Phi^l ... \Phi^{L-1}, \Phi^L\} \tag{15}$$

These sets of local tasks have to be coordinated to achieve the global solution. The coordinator, as an independent task, will have its own target function. Taking everything into account, this concept is the base on which one may build the new scheme of the ANN learning algorithm's structure (Fig. 3). It is the neural network 's and the learning algorithm's hierarchical structure. The two-level ANN learning algorithm can be described as a set of procedures. The procedures on the first level are responsible for solving their local tasks and calculating the

part of matrix weight coefficients. The second-level procedure has to coordinate all the local procedures (tasks) using its own local target function. The third-level procedure calculates the learning parameters which are used by the second-level. There is a vertical decomposition and interaction between the procedures. Two types of information are sent between the levels. From the second level to the first level, one is a downward transmission of control signals:

$$\Gamma = (\Gamma^1, \Gamma^2, ...\Gamma^{L-1}) \tag{16}$$

Where:
Γ^l - vector of data sent from the coordinator to the two neighboring sub-networks l and $l+1$,
$l = 1 \div (L-1)$ - number of sub-networks,

From the first level to the second level two sets of feedback signals are sent:

- forward feedback errors, which are generated by every sub-network when all sub-networks calculate their local target functions value Φ^l:

$$\mathcal{E}_F = (\mathcal{E}_F^1, \mathcal{E}_F^2, ...\mathcal{E}_F^{L-1}) \tag{17}$$

- backward feedback errors, which are calculated by every sub-network in the back propagation procedure:

$$\mathcal{E}_B = (\mathcal{E}_B^1, \mathcal{E}_B^2, ...\mathcal{E}_B^{L}) \tag{18}$$

2.1 Levels of calculation complexity

The standard ANN learning algorithm is a non-linear minimization task without constraints. To solve this task, iteration procedures are used. Using the most popular back propagation algorithm for standard multilayer network's structure, one has to choose a lot of control parameters. The algorithm is time-consuming and its convergence is not fast. Dividing the primary algorithm into the sub-network tasks, the local target functions are simpler and can be used in different procedures. Additionally, a new procedure is needed: the coordination procedure. In practice, however, the coordinator does not have the ability to find all the parameters needed for the first-level procedures. To solve this problem, a multi-level decision hierarchy is proposed [1]. The problems is solved by the iteration algorithm on both the first and the second level. One can observe specific dynamic processes. These processes are non-linear and use a lot of control parameters too. During the learning process these parameters are stable and do not change. Practice proves that this solution is not optimal. To control the way learning parameters are changed in the iteration process, an additional level could be used - the adaptation level (Fig. 3). Thus, one can build three levels as a minimum:

- The local optimization procedures: the algorithm is defined directly as a minimization task without constraints.
- The coordination procedure: this algorithm could be defined directly as a minimization of the target function as well. Constraints could exist or not.
- The adaptation procedure: the task or procedure on this level should specify the value of learning parameters not only for the coordinator level, but also on the first level. To solve this task, a procedure should achieve dynamic characteristic of the learning process from all the levels.

As a conclusion, one can state that the complexity of the problem increases from the first level to the next one. The coordination and adaptation procedures need time to solve their own tasks.

3 Calculation algorithm structure

The deep neural network with an input layer, a set of hidden layers and an output layer can be used for further considerations. As standard $ReLu$ activation function is used for all the hidden layers. For an output layer both sigmoid or $ReLu$ activation functions could be used. A three parts learning algorithm will be considered to decompose the standard learning algorithm's structure into sub-network tasks as shown on Fig. 3.

3.1 Forward calculation

All sub - networks are independent because the coordinator sent an input and an output vector Γ^l for all the layers. Sub-networks can calculate all values in a parallel way:

- For the first sub-network:

$$\Phi^1(W^1, X^0, \Gamma^1) = \frac{1}{2} \cdot (X^1 - \Gamma^1)^T \cdot (X^1 - \Gamma^1) \tag{19}$$

Others relations:

$$X^1 = F(U^1) \tag{20}$$

$$U^1 = W^1 \cdot X^0 \tag{21}$$

Where: X^0, X^1 - input and output vector of the first layer, accordingly,
F - vector of activation function. $ReLu = max(0, U^1)$,
U^1 - internal vector.

- For the all hidden layers $l = 2 : (L - 1)$:

$$\Phi^l(W^1, \Gamma^{l-1}, \Gamma^l) = \frac{1}{2} \cdot (X^l - \Gamma^l)^T \cdot (X^l - \Gamma^l) \tag{22}$$

$$X^l = F(U^l) \tag{23}$$

$$U^l = W^l \cdot \Gamma^{l-1} \tag{24}$$

Where:
X^{l-1}, X^l - input and output vector of all the hidden layers, accordingly,
F - vector of activation function. $ReLu = max(0, U^l)$,
U^1 - internal vector.

- For the output layer:

$$\Phi^L(W^L, \Gamma^{L-1}, Y) = \frac{1}{2} \cdot (X^L - Y)^T \cdot (X^L - Y) \tag{25}$$

$$X^L = F(U^L) \tag{26}$$

$$U^L = W^L \cdot \Gamma^{L-1} \tag{27}$$

Where:
X^L - output vector,
Y - learning vector P epoch included.

3.2 Backward calculation

When all sub-networks have finished the forward calculation process, the next step can begin, that is the backward calculation. The calculated error will be sent to the coordinator. Modified formulas from subsection 1.1 are used. All subnetworks with their own target functions can calculate their own backward errors in a parallel way:

- For the first sub-network, only the forward error is calculated,(see Fig.4):

$$\mathcal{E}_F^1 = X^1 \tag{28}$$

From formula(19) partial derivatives for the matrix of weight coefficient W^1 are:

$$\frac{\partial \Phi^1}{\partial W^1} = \frac{\partial \Phi^1}{\partial X^1} \cdot \frac{\partial X^1}{\partial U^1} \cdot \frac{\partial U^1}{\partial W^1} \tag{29}$$

$$\frac{\partial \Phi^1}{\partial X^1} = E^1 = X^1 - \Gamma^1 \tag{30}$$

Formula (29) can be rewritten in the matrix form:

$$\frac{\partial \Phi^1}{\partial W^1} = \{E^1 \odot 1(U^1)\} \cdot (X^0)^T \tag{31}$$

Where:
$1(U^1)$ - derivative of the $ReLu$ activation function $ReLu^{'} = max(0, U^1)^{'} = 1(U^1)$.
The first sub-network can calculate a new value of the matrix weight coefficients W^1:

$$W^1(n+1) = W^1(n) - \alpha \cdot \frac{\partial \Phi^1}{\partial W^1} \tag{32}$$

Where:
n - current iteration number.

- For all the hidden layer $l = 2 : (L-1)$ and using formula (22), partial derivatives for the matrix of weight coefficient W^l and an input vector from the coordinator Γ^{l-1} are:

$$\frac{\partial \Phi^l}{\partial \Gamma^{l-1}} = \frac{\partial \Phi^l}{\partial X^l} \cdot \frac{\partial X^l}{\partial U^l} \cdot \frac{\partial U^l}{\partial \Gamma^{l-1}} \tag{33}$$

$$\frac{\partial \Phi^l}{\partial X^l} = E^l = X^l - \Gamma^l \tag{34}$$

$$\frac{\partial U^l}{\partial \Gamma^{l-1}} = W^l \tag{35}$$

Formula (33) can be rewritten in the matrix form:

$$\frac{\partial \Phi^l}{\partial \Gamma^{l-1}} = \mathcal{E}^{l-1} = (W^l)^T \cdot \{1(U)^l \odot \mathcal{E}^l\} \tag{36}$$

Where:
$1(U^l)$ - derivative of the $ReLu$ activation function $ReLu^{'} = max(0, U^l)^{'} = 1(U^l)$,

$$\mathcal{E}_B^{l-1} = \Gamma^l - \beta \cdot \frac{\partial \Phi^l}{\partial \Gamma^{l-1}} = \Gamma^l - \beta \cdot \mathcal{E}^{l-1} \tag{37}$$

The derivative for the weight coefficients of matrix W^l is:

$$\frac{\partial \Phi^l}{\partial W^l} = \frac{\partial \Phi^l}{\partial X^l} \cdot \frac{\partial X^l}{\partial U^l} \cdot \frac{\partial U^l}{\partial W^l} \tag{38}$$

$$\frac{\partial U^l}{\partial W^l} = \Gamma^{l-1} \tag{39}$$

Formula (38) can be rewritten i the matrix form:

$$\frac{\partial \Phi^l}{\partial W^l} = \{E^l \odot 1(U^l)\} \cdot (\Gamma^{l-1})^T \tag{40}$$

A sub-network can calculate the new value of the matrix weight coefficients:

$$W^l(n+1) = W^l(n) - \alpha \cdot \frac{\partial \Phi^l}{\partial W^l} \tag{41}$$

Where:
n - current iteration number.

- For the output layer using formula (25) partial derivatives for the matrix of weight coefficient W^L and an input vector from the coordinator Γ^{l-1} are:

$$\frac{\partial \Phi^L}{\partial \Gamma^{L-1}} = \frac{\partial \Phi^L}{\partial X^L} \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial \Gamma^{L-1}} \tag{42}$$

$$\frac{\partial \Phi^L}{\partial X^L} = \mathcal{E}^L = X^L - Y \tag{43}$$

$$\frac{\partial U^L}{\partial \Gamma^{L-1}} = W^L \tag{44}$$

Formula (42) can be rewritten in the matrix form:

$$\frac{\partial \Phi^L}{\partial \Gamma^{L-1}} = \mathcal{E}^{L-1} = (W^L)^T \cdot \{1(U)^L \odot \mathcal{E}^L\} \tag{45}$$

$$\mathcal{E}_B^{l-1} = \Gamma^{L-1} - \beta \cdot \frac{\partial \Phi^L}{\partial \Gamma^{L-1}} = \Gamma^{L-1} - \beta \cdot \mathcal{E}^{L-1} \tag{46}$$

Applying the same procedure as above, a set of formulas is written:

$$\frac{\partial \Phi^L}{\partial W^L} = \frac{\partial \Phi^l}{\partial X^L} \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial U^L}{\partial W^L} \tag{47}$$

$$\frac{\partial U^L}{\partial W^L} = \Gamma^{L-1} \tag{48}$$

Formula (47) in the matrix form:

$$\frac{\partial \Phi^L}{\partial W^L} = \{\mathcal{E}^L \odot 1(U^L)\} \cdot (\Gamma^{L-1})^T \tag{49}$$

The output sub-network can calculate the new value of the matrix weight coefficients:

$$W^L(n+1) = W^L(n) - \alpha \cdot \frac{\partial \Phi^L}{\partial W^L} \tag{50}$$

Where:
n - current iteration number.

When all sub-networks have finished calculation of their local target functions, the forward $\mathcal{E}^l_F$ and backward $\mathcal{E}^l_B$ feedback information are sent to the coordinator. At the moment, all the sub- networks modified their matrixes of weight coefficient W^l for $l = 1 : L$.

4 Coordinator structure

In a hierarchical learning algorithm, the coordinator plays the main role. It is now time to decide what kind of coordination principle will be chosen. This principle specifies various strategies for the coordinator and determines the structure of the coordinator. In [1] three methods were introduced in which the interaction could be performed:

- Interaction Prediction. The coordination input may involve a prediction of the interface input. This principle can be used for farther consideration.
- Interaction Decoupling. Each first-level sub-system is introduced into the solution of its task and can treat the interface input as an additional decision variable to be free. Consequently, sub-systems are completely decoupled.
- Interaction Estimation. The coordinator specifies the ranges of interface inputs over which they may vary.

For a deep neural network containing L layers, the coordinator prepare $l = 1 \div (L-1)$ coordination signal Γ^l. One of them is treated as input vectors and the other as learning data vectors for local target functions Φ^l for $l = 1 : (L-1)$. Every coordinator signal is correlated with two feedback signals $\mathcal{E}^l_F$ and $\mathcal{E}^l_B$. These signals are calculated by the first level sub-networks and are sent to coordinator. The coordinator uses its own target function:

$$\Psi = \frac{1}{2} \sum_{i=1}^{i=L-1} \{(\Gamma^i - \mathcal{E}^i_B)^T \cdot (\Gamma^i - \mathcal{E}^i_B)\} + \{(\Gamma^i - \mathcal{E}^i_F)^T \cdot (\Gamma^i - \mathcal{E}^i_F)\} \tag{51}$$

To minimize this function, the gradient method is used:

$$\frac{\partial \Psi}{\partial \Gamma^i} = (\Gamma^i - \mathcal{E}_B^i) + (\Gamma^i - \mathcal{E}_F^i) = 2 \cdot \Gamma^i - [\mathcal{E}_B^i + \mathcal{E}_F^i] \tag{52}$$

The new value of coordination signal is calculated using the gradient method:

$$\Gamma^i(n+1) = \Gamma^i(n) - \rho \cdot \frac{\partial \Psi}{\partial \Gamma^i} \tag{53}$$

This new coordinator signals value is sent to the first level sub-networks and the entire process is repeated. In formula (53) new hyper-parameter ρ is used. Its value will be defined by adaptation level.

5 Adaptation level

In formula (53) the learning parameter ρ could be constant or could change during the iteration process. In the beginning the learning process is very dynamic, a large oscillation could be seen in the first level of the local target functions. Because parameter ρ has to be small, the iteration process is stable. However, although the minimum Ψ is achieved asymptotically, it happens very slowly. To improve this algorithm, the coordinator sends the target function value $\Psi(n)$ to the adaptation level. The following simple algorithm could be used:
If $\Psi(n) > \Psi(n-1)$ then $\rho = k_d \cdot \rho$, where: $k_d = 0.95 \div 0.98$ - decreasing parameter,
If $\Psi(n) < \Psi(n-1)$ then $\rho = k_i \cdot \rho$, where: $k_i = 1.02 \div 1.05$ - increasing parameter. The new ρ is sent to the coordinator and used in the iteration process. To make the entire learning process more stable, Ψ is usually averaged by epoch:

$$\Psi(n, p+1) = \frac{1}{N_p} \cdot \Psi(n,p) + \frac{N_p - 1}{N_p} \cdot \sum_{p=1}^{p=N_p-1} \Psi(n,i) \tag{54}$$

Where: N_p - number of input vector in epoch, p - actual index value.

6 Numerical example and conclusion

In a life insurance company the underwriting process has been playing the central role in risk control and premium calculation. A deep neural network could be used to help the insurance agents to classify the insurance applicant and calculate the first version of premium. Therefore, a special short questionnaire was prepared which includes only 10 main questions (Fig.4). All the data were divided into three subsets: - learning set includes 250 records, -verification set includes 50 vectors,- testing includes 100 vectors.Network is trained on single data-points. An epoch includes 250 iterations (one input and output data = one iteration). Beginning values of the learning coefficients (hyper-parameters): $\rho_0 = 0.01$, $\alpha_0^l = 0.01$, $\beta_0^l = 0.01$ for l=1-L, $\gamma_0 = 0.02$

Dynamic characteristics of the second subnetwork contain two phases. From start to 4000 iteration, error decrease very fast. In the next phase, the learning process is not optimal. The learning process includes 40000 single iterations.

In Fig. 6. the feedback signals do not have optimal characteristics. Probably the learning coefficient ρ is too hight and the coordinator tries to accelerate the learning process. Result is reverse. The adaptation level should force its own strategy to stabilize the learning process. Future works should focus on the coordinator and the adaptation level strategy. The main question is - how to improve the characteristics of the learning process?

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
Nr	Sex	Years old	Weight in kg	Growth in cm	Smoking	Father's long life	Mather's long life	Children number	Living condition	Living place
1	1	48	70	190	0	68	67	0	1	2
2	1	62	71	182	0	82	91	1	0	0
3	0	64	83	180	5	89	100	2	1	2
4	1	40	107	177	10	74	66	2	1	2
5	1	58	120	183	20	90	77	3	1	1

Figure 4: An input data structure example

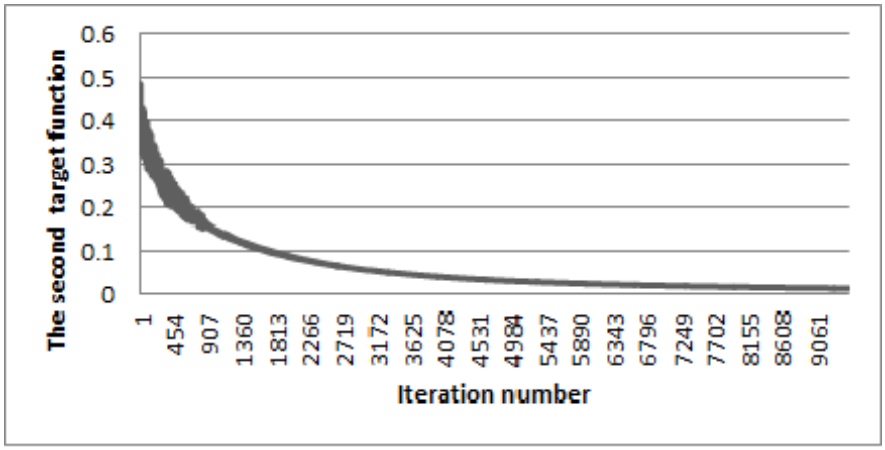

Figure 5: Dynamic characteristic of the second target function Φ^2

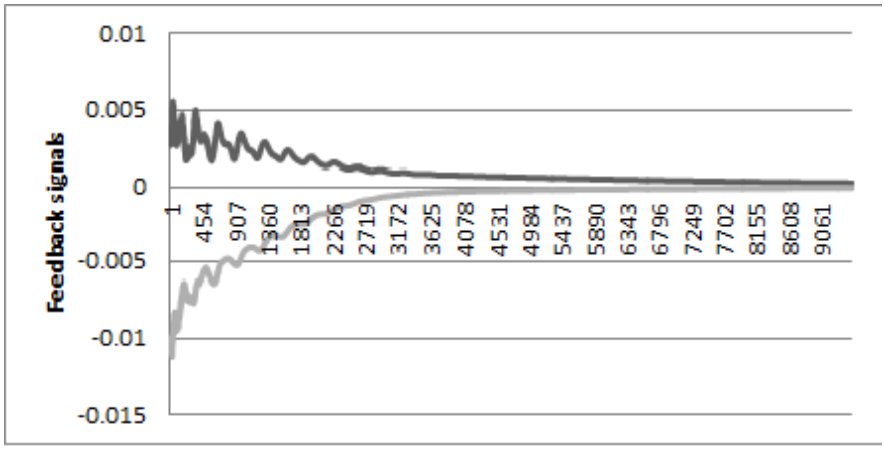

Figure 6: Dynamic characteristics of the two feedback signals $\mathcal{E}_B^2$ (black) and $\mathcal{E}_F^2$ (gray)

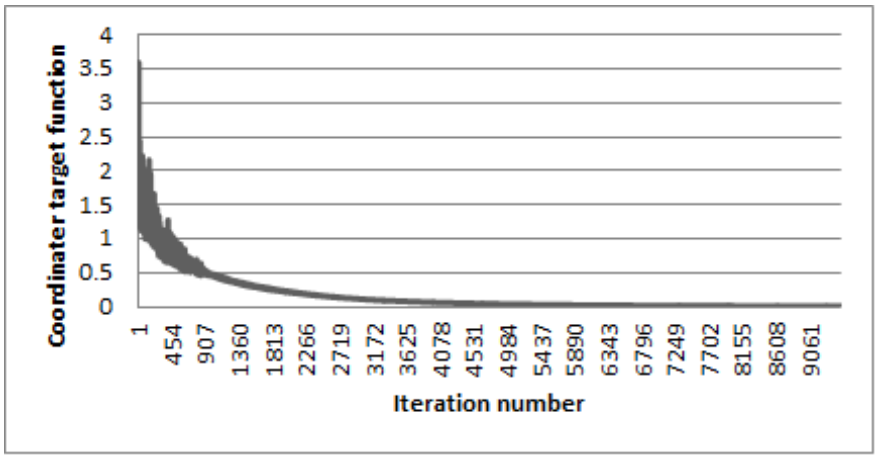

Figure 7: Dynamic characteristics of the coordinator

References

[1] M. D. Mesarocic, D. Macko, and Y. Takahara, Theory of hierarchical multilevel systems, Academic Press, New York and London, 1970.

[2] Ch. M. Bishop, Pattern Recognition and Machine Learning, Springer Science + Business Media, LLC 2006.

[3] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016

[4] Zeng-Guang Hou, Madan M. Gupta, Peter N. Nikiforuk, Min Tan, and Long Cheng, "A Recurrent Neural Network for Hierarchical Control of Interconnected Dynamic Systems", IEEE Transactions on Neural Networks, Vol. 18, No. 2, March 2007.

[5] Joarder Kamruzzaman, Rezaul Begg, Artificial Neural Network in Finance and Manufacturing, Idea Group Publishing, Hershley, Pennsylvania 2006.

[6] Bijaya Adhikari, Yao Zhang, Aditya Bharadwaj, B.Aditya Prakash, Condensing Temporal Network using Propagation.

[7] Suraj Srinivas,Ravi K. Sarvadevabhatla, and others, An Introduction to Deep Convolution Neural Nets for Computer Vision, Academic Press 2017.

[8] S.K. Zhou, H. Greenspan, D. Shen, Deep Learning for Medical Image Analysis, Academic Press 2017

[9] M. Nielson, Neural Network and Deep Learning, Determination Press, 2015

[10] S. Placzek, B. Adhikari, "Analysis of Multilayer Neural Network with Direct Connection Cross-forward Connection", CS&P Conference 2013, The University of Warsaw, Warsaw 2013.

[11] L. Rutkowski, Metody i techniki sztucznej inteligencji, Wydawnictwo Naukowe PWN, Warszawa 2006.