ECONTECHMOD. AN INTERNATIONAL QUARTERLY JOURNAL – 2018, Vol. 08, No. 2, 9-14

Building computer vision systems using machine learning algorithms

N. Boyko , B. Dokhniak, V. Korkishko

Lviv Polytechnic National University, Lviv, Uktaine; e-mail: nataliya.i.boyko@lpnu.ua

Received February 19.2018: accepted May 20.2018

Abstract. This article is devoted to the algorithm of training with reinforcement (reinforcement learning). This article will cover various modifications of the Q-Learning algorithm, along with its techniques, which can accelerate learning using neural networks. We also talk about different ways of approximating the tables of this algorithm, consider its implementation in the code and analyze its behavior in different environments. We set the optimal parameters for its implementation, and we will evaluate its performance in two parameters: the number of necessary neural network weight corrections and quality of training.

Keywords: Training with reinforcement, Q-Learning, Neural networks, Markov environment.

INTRODUCTION

Watkinson proposed the Q-Learning algorithm in 1989. This algorithm relates to a group of training algorithms with reinforcements. Learning with reinforcements represents a class of tasks in which the agent, acting in a particular environment, must find the optimal strategy for interaction with it. One of the popular methods for solving such problems is the Q-Learning algorithm. The agent's training information is presented in the form of a "reward", which has a certain number of values for each agent transition from one state to another. No other additional information for training the agent is provided. An important property of the Q-Learning algorithm is the ability to use it even in cases where the agent has no prior knowledge of the environment in which it will be located.

When working with the Q-Learning algorithm, creation of a table for the function of the estimation of state-activity pairs is created. One of the conditions for the convergence of the algorithm in the case of using the tabular representation of the Q-values function is a multiple test of all possible state-activity pairs. Practical tasks usually have a large number state-activity pairs, which makes it impossible for tabular Q-Learning to solve problems of this type. In order to solve this problem it is necessary to use the approximation of the table of Q-values. One means of effectively approximating the table of Q-values is the use of multilayer perceptron. It is this method that is devoted to this work.

Purpose of paper: exploring opportunities and perspectives of using the algorithm of training with reinforcement with the help of neural networks.

Q-LEARNING ALGORITHM AND ITS MODIFICATIONS

The problem of reinforcement learning in general formed as follows. For each transition from one state to another appointed some scalar value "reward." The system receives a "reward" when making transition. The purpose of the system is a management policy that maximizes the expected amount of compensation known as a return. The function of the value is the prediction value of all states

$$V(xt) \leftarrow E\{\sum_{k=0}^{\infty} y^k \, r_{t+k}\}$$

where rt – the award received during transition from the system in state x_t to x_{t+1} and γ - discount factor ($0 \leq \gamma \leq 1$). Thus, $V_{(xt)}$ represents the discount amount of rewards system can get at time t. This amount depends on the selected sequence of actions defined by policy management. The system needs to find a management policy that maximizes $V_{(xt)}$ for each state.

Q-Learning Algorithm is not working with the function of value and uses instead of it Q-function argument which is not only the state but also action. It is possible to present Q-function and thereby find the optimal management policy (policy). This statement looks like: [6-7]

$$Q(x_t, a_t) \leftarrow r_t + y * V(x_{t+1})$$

where at –action selected at time t from the set of all possible actions A. Since the aim of the system is the total reward maximization, we create replacement max $Q(^{x_{t+1}}, a)$ and then we get the following:

$$Q(x_t, a_t) \leftarrow r_t + y \; max(x_{t+1}, a)$$

Values are stored in a 2-dimensional table, the entry submitted by states and actions. The tabular representation of Q-function and Markov environment creates an element of convergence of the algorithm Q-Learning.

Systems that use this algorithm, usually combined with a time difference (TD (λ)), which was suggested by Sutton. If the time difference method (λ) is equal to 0, it means that the upgrade involves only the current and the following values forecast Q-values. Therefore, in this case, a method is called one-step Q-Learning. Expression of this algorithm is as follows:

$$Q(x_t,a_t) \leftarrow Q(x_t,a_t) + a(r_t + y * maxQ(x_{t+1}) \_ Q(x_t,a_t))$$

Analyzing expression of the Q-update function can conclude that the maximum use of this function is not good usually. In the early stages of learning algorithm Q-value table contains estimates that are not ideal, and even in the later stages usage of maximum could lead to a revaluation of Q-values. Moreover rule of updating algorithm Q-Learning in combination with the time difference needs zero value for λ in choosing actions based on "not greedy" policy (policy action in which selected from some probability that depends on the value Q-functions for the state, as opposed to "greedy" when the selected action with the largest Q-value). These deficiencies have caused modification algorithm Q-Learning, which is one of the sources called SARSA (State-Action-Reward-State-Action), the other - modified Q-Learning. The main difference between these algorithms is that updates the rules Q-max values deleted operator. As a result, it is guaranteed that the error of the time difference will be calculated correctly whether the action will be selected according to the "greedy policies" or not, without having to reset the time difference. If the action will be selected in accordance with the "greedy politics", then this rule will fully comply update updates the formula above.

Peng and Williams in their work in one-step algorithm Q-Learning introduced another method of combining Q-Learning and the time difference, called Q (λ). This method is based on performance of the conventional one-step update policies to improve the prognosis of this Q and subsequent use of the time difference between arranged next to each other "greedy" predictions. Thus, this method does not depend on which policy has been selected. [4]

METHODS APPROXIMATION Q - VALUES.

One of the easiest ways of dealing with a large volume dimension which will run the agent is sampling, i.e. partitioning state space for small area each table entry field is Q - values. Using this approach received a gross generalization states. The success of this event depends on how efficient this you have to partition function Q - values. On the one hand for high accuracy it is required to do more partitions on a small area and, consequently, use the table Q - values greater volume, resulting in a need for a larger number of updates during training. On the other hand, splitting a volume area can lead to the inability to achieve optimal management policy.

Thus, this method is the problem-oriented and requires a lot of effort for selection of optimal partitioning.

There are methods that can accelerate the learning process by using tables Q - values large. One such method is the method of Hamming distance. Using this method all classes are given in the binary form and given threshold similarity (the number of bits in which each state can be different from the other). When the correction Q - valuesat the same time is updated for the selected state and for all the states to which the Hamming distance is selected less than a given threshold. Thus, the accelerated spread of Q - values in the table.

Method CMAC (Cerebellar Model Articulator Controller), proposed by the Albus (Albus) is a compromise between using ordinary table Q-values and approximation of continuous functions. This method is known in the literature as a "tile" coding (tiles coding). Approximation CMAC structure consists of several layers. Each layer is divided into intervals of equal length ("tiles") using the quantization. As each layer has a quantum function, the "tiles" layers are shifted relative to each other. Thus, system status, filed at the entrance CMAC, is associated with a set that overlap shifted tiles. The weighted sum of the indices of tiles and gives the output value. Method SMAS had success in solving difficult problems with continual space values, including the task of robot control. But nevertheless, despite the successful use, this algorithm requires quite difficult settings. The accuracy of functions that approximates is limited expansion of quantization. High precision requires a larger number of quantization scales and a long-term study environment. [7-8]

RBF networks (Radial Bases Functions) closely related to the CMAC and conventional tables. When using this method of approximation table instead of the table of Q - values stored Gaussian table of functions or quadratic function. Rolling systems passed through all the functions of the function then summed and as a result, we obtain approximate values. [3]

Consequently, all of the above methods have a common drawback - poor scalability when working with multidimensional space. If the present system with i-inputs that need for quality approximation N basic functions, that means that we the need N^t basic functions. Thus, the number of basic functions grows exponentially regardless of the dimension of the input vector.

Later in this article, a paragraph about the robot control we will consider this method as static approximation cluster analysis. When using this method, each step is connected with multiple clusters that provide assessments of defined class situations. During the upgrade evaluation of Q - values for the current state arises update all states that belong to this cluster. The authors of this method set limitation for this method: the difficulty settings to create semantic clusters and value that the cluster once formed cannot be broken in the future.

It is known that multilayer perceptron is a good approximation function and, of course, there is a theoretical explanation. There is Kolmogorov theorem of mapping neural networks (Kolmogorov Mapping Neural Network Existence Theorem), which states that the directly distributed neural network with three layers (input layer, hidden layer, and output layer) can accurately represent any continuous function. The work of Lin is one of the first works in which in order for table to approximate and table with Q - values is used multilayer perceptron. Using a neural network to approximate Q - function has the following advantages:

— Effective scaling for space has more dimensions;

— Generalization for large and continuous state space;
— The possibility of implementation of parallel hardware.

FEATURES OF NEURAL NETWORKS IN PROBLEMS OF REINFORCEMENT LEARNING

When working with neural networks we distinguish two areas of training: learning with a teacher (supervisor learning) and learning without a teacher (unsupervised learning). The algorithms of reinforcement learning don`t belong to the above areas and the use of multilayer perceptron for approximation of functions in problems of reinforcement learning differs from conventional supervised learning. Comparing normal usage of perceptron for approximation problems and its use as part of the reinforcement algorithm can distinguish two main points:

— In approximation problems normal training is provided in some training set whose elements are constantly repeated. When learning reinforcement there is no preset training set. Input samples are formed by the interaction of agent with the environment, and thus in learning some designs are more common than others. But when dealing with the continuous environment – there is high probability that the input pattern will be met only once.
— In approximation problems normal training is conducted on known results, known true values of approximating function in certain points, which is not a reinforcement learning, and learning takes place on estimates Q-values which gradually change in the learning process.

CONNECTABLE Q-LEARNING

When using connect – approach in the Q-Learning algorithm tabular representation of Q-functions become replaced by neural network. The input of the network is fed by conditions, and initial data is estimated by Q-values. Thus, no major changes in the classic Q-Learning are not included, except mechanism of changes in estimates storage of Q-values. This article uses a method of neural network offered by Lin which consists of applying a separate neural network for each action. [9]

Q value of a_n action

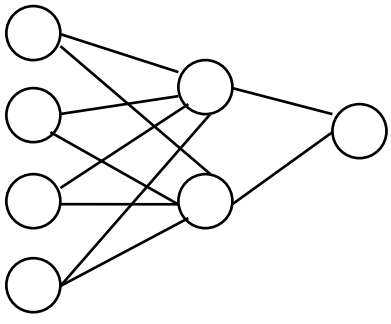

Fig.1. Q-approximation function using a plurality of neural networks

At each iteration of the algorithm, current state is fed to the inputs of each neural network, but updating the weights is performed only for one neural network, whose actions were selected. When using a one-step Q-Learning, error correcting network weights is

$$r_t + y * maxQ(x_{t+1,}a_{t+1}) - Q\ (x_t, a_t)$$

It is worth noting that Lin in his work used a special method of correction weights of the neural network, named backward replay. When using this method, the weights of the neural network are updated only when you reach absorbing system state (final state, for example, when reached any purpose). Using this technique provides storing all pairs of state - action that meets the system before reaching the absorbing state. Algorithm updates using the classic method of Q- values from reverse repetition:

To reverse repetition:

$$\{(x_{0,}a_{0,}x_{1,}r_0\ ) \dots (x_{n,}a_{n,}x_{n+1,}r_n\ )\}$$

Perform

1. $t \leftarrow n$
2. $e_t \leftarrow Q(x_t, a_t)$
3. $u_{t+1} \leftarrow Max\{Q(x_{t+1}, k\ )|\ k\ \in \mathrm{A})\}$
4. $e'_t \leftarrow\ r_t +\ \gamma * [(1-\lambda) * u_{t+1} + \lambda * e_{t+1}]$

Then, you configure the network that implements the algorithm using reverse distribution, where the error is approximately equal $(\mathrm{x_t},\ \mathrm{a_t})\mathrm{e'_t} - \mathrm{e_t}$

If t = 0 output, otherwise t ← t- 1; transition to step 2.

The idea of the methodology used Lin, lies in the fact that a correct assessment Q- values is known only at achieving system absorbing state. In this case, the Q-rating values equal prize. When removing from absorbing state evaluation Q-value is reduced by using discount factor. Scrolling the list of state - in reverse allows you to perform studies with a more precise estimate. However, the sequence of steps performed by the system may be sub-optimal and therefore assess which training will be carried out, as will be optimum. To address this shortcoming in their methods Lin used a weighted sum consisting of two components:

1)Current grades of Q-function
2)Grade obtained using the recursive expression in step 4 of the algorithm.

Parameter λ, used in step 4 determines which of these two components need to be provided with more benefits.

When using modified Q- Learning expression in step 3 to be replaced Q ($u_{t+1} \leftarrow Q(x_{t+1}, a_{t+1})_{\lambda}$) Algorithm requires making more serious changes that can be implemented as follows:

1) Add steps 2a and 4a, which are as follows

2a: $e2_t \leftarrow Max\{Q(x_t, k\ )|\ k\ \in |\mathrm{A}|\}$

4a: $e2_t \leftarrow r_t +\ \gamma \cdot u_{t+1}$

2) Error in step 5 will be as follows:

$$e'_t - e2_t + e2'_t - e_t$$

The obvious drawback of reverse repetition technique is the need to store information about all the passages that

have been implemented the system before reaching the absorbing state.

Also in the work of Sutton is described the use of algorithm TD by neural networks. This algorithm allows obtaining good results without storing lists of long pairs of condition - action. The basis of this algorithm is vectoring "traces of conformity" which provided the weights of the neural network. Using "tracks matching" allows updating the weights take into account the error in the previous steps, as they remain the weighted sum of the ingredients weekend. Version of this algorithm adapted and modified for Q-Learning and Q (λ) below: [1-2]

modified Connectable Q-Learning

Set "tracks matching" zero, e0 = 0

t = 0

Choose action, at

If t > 0, we correct weights:

$wt = wt_{-1} + \alpha \cdot (rt_{-1} + \gamma \cdot Qt - Qt_{-1}) \cdot et_{-1}$

Calculate initial gradient $\nabla_w Qt$ only for that network whose actions were selected.

$et = \nabla_w Qt + \gamma . \lambda \cdot et_{-1}$

Take action and get at "award" rt

If the absorbing state is reached, then we end; otherwise

$t \leftarrow t + 1$ and transition to step 3.

Connectable Q-Learning for Q (λ)

Set "tracks matching" zero, $e_0 = 0$

t = 0

Choose action, at

If t> 0, then be corrected weights:

$$w_{t-1} = w_{t-1} + \alpha \cdot ([r_{t-1} + \gamma \cdot \max Q_t - Q_{t-1}] \cdot \nabla_w Q_{t-1} + [r_{t-1} + \gamma \cdot \max Q_t - Q_{t-1}] \cdot e_{t-1})$$

$et = \nabla_w Qt + \gamma \cdot \lambda \cdot et_{-1}$

Calculate the input ingredients $\nabla_w Qt$ only for one network, the effect of which has been selected.

Take action a_t and "to receive the award» r_t

If the absorbing state is reached - we end, and if not,

$t \leftarrow t + 1$ and transition to the 3rd step.

When using MCQ-L should be stored neural networks weight "footprints matching" last mentioned Q-function Q and reward r. For Q (λ) storage costs more, as addition is necessary to keep the output gradients between steps neural network algorithm. Despite this it costs significantly less than those costs which are necessary for keeping the list of state-action methods using reverse repetition.

OVERVIEW OF TASKS FOR ROBOT CONTROL

The problem of robot control in 2-dimensional space has been solving at different times by different methods. Most work in this area dedicated to planning routes, which analyzed the environment with the aim of finding the most convenient way. One of the first works in this field is the work of Wilson.

This work led to the birth of a whole class of problems dealing animates (ANIMAT = ANIMAL + ROBOT), that works enrolled using an algorithm of reinforcement. Classic animat of Wilson works in the discrete world and continually trains in the same environment. Animata's purpose – is to learn how to achieve the goals of any training position for a minimum number of steps.

All these works are characterized by the fact that tuition is always done in the same environment, so the warranty is only that the robot can only operate effectively in this environment and it is unknown how effective will be his behavior with the little change of the environment. In our work, the experiments with the robot, which learns many pretty typical examples, robot can function effectively, got a completely unfamiliar environment.

TASK MANAGEMENT ROBOT

Efficiency of described algorithms in this work analyzed using developed software simulator, operating in the 2-dimensional continuous medium. There was a task for a robot - to reach the goal, not facing any obstacles. Approximate scheme of sensors and work:

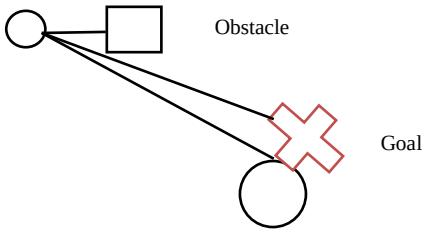

Fig.2. Recognition Scheme interference.

The learning process has been divided into trials. At each stage of education robot and goal were moved to a new access point, which took place after the generation of the new location of obstacles. Robot received award only in the end of the stages, in all other cases, the reward was zero. The stage ended in achieving the goals robot and receiving awards in the form of units. In this article, we reviewed the case where the robot could not reach the goal because of the location of obstacles. This case will be described and listed below.

We generated 26 randomly placed on the map, and there is a way in which the robot reaches the target (as shown below).

Processes of exploration and exploitation are important in the algorithms of reinforcement learning. The first step is to explore how you can set the environment by choosing less priority action. The final step is to go directly to the operation embodied in this article using the Boltzmann distribution:

$$P(a_t|x_t) = \frac{\exp(Q(x_t, a_t)/T}{\sum_{a \in A} \exp(Q(x_t, a)/T)},$$

where T - the temperature that regulates the degree of randomness of selection with the largest Q- value.

Results: In an experiment with systems, the main component, neural network consists of, the problem is related to the fact that reducing the error rate networks strongly depends on init weights. Therefore, before conducting our experiments, we have carefully studied all the possible distribution of weights for better performance of our neural network.

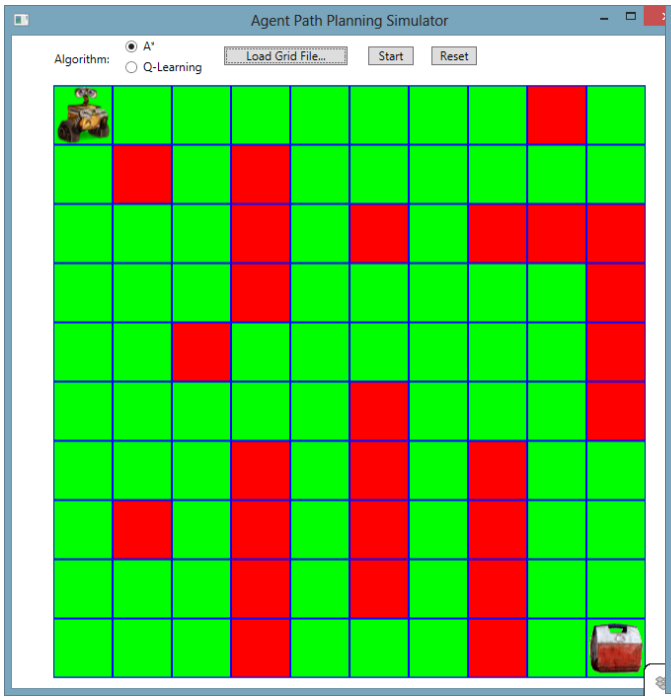

Fig. 3. Generated Map interference.

Besides using Q-Learning algorithm we should pay attention to changes in temperature interval T and the speed of change, because algorithm depends on these parameters convergence. Because these parameters and the "cornerstones" were considered:

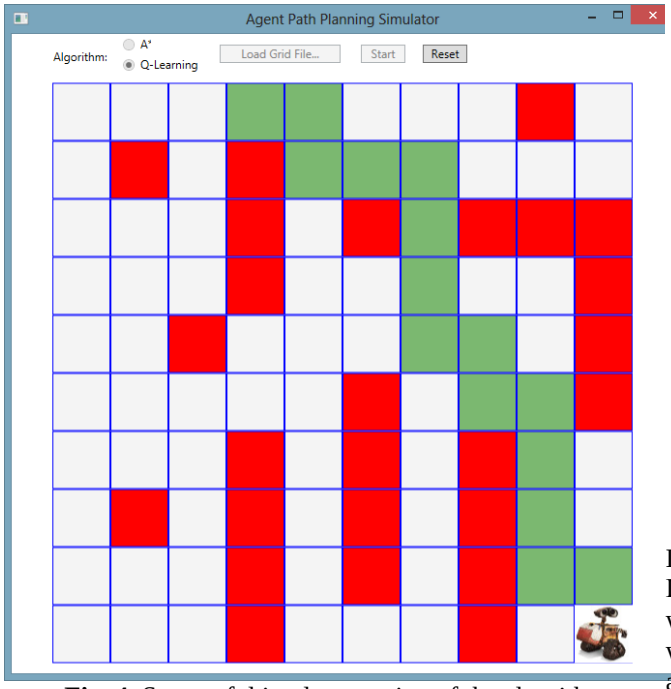

Fig. 4. Successful implementation of the algorithm.

As we see in the picture above, the algorithm remembers the way from any cell on the map to the target and then just executes it. However, we should also consider the case when the target will be "locked" by obstacles, such as the unpredictable situation behave neural network of reinforcement.

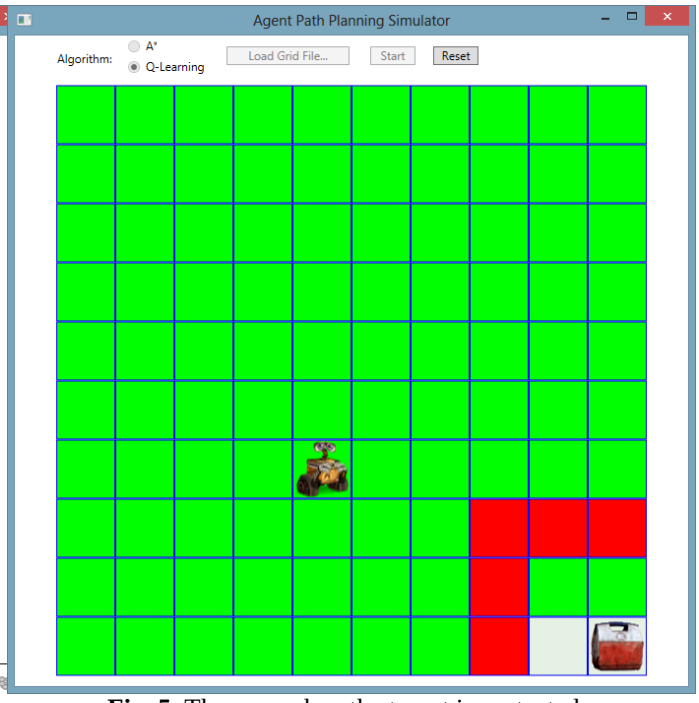

Fig. 5. The case when the target is protected

In this case, the neural network and the algorithm simply loop. The algorithm tries to sort through all the possible admission to the goal, but each time facing one and the same obstacles. Since our robot stops only when the target is reached, program simply loops.

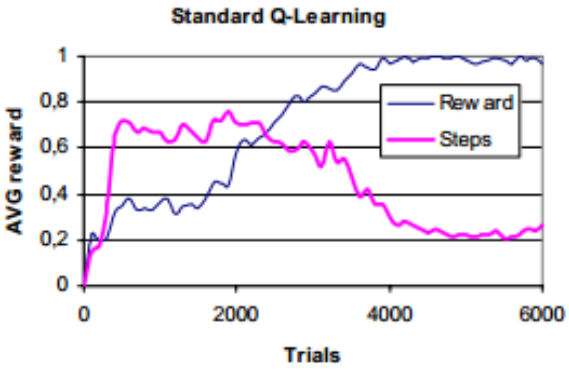

Fig. 6. Charts comparing two methods: interactive update and feedback repetition

CONCLUSION

We presented a description of the algorithm Q-Learning and its various options (such as modification Q-Learning IQ (λ). Each of the above-mentioned algorithms was presented separately from two sides (direct correction weights and reverse playback). Experiments and practical application can be seen in the work and in our created environment. The researchers created the optimal conditions of algorithm parameters and compared their effectiveness.

So, after all that is said and analyzed above, we can conclude that the best and most effective algorithm was modified Q-Learning method with an immediate change of scale. This algorithm makes gains in many criteria such as quality, speed and memory cost. It is also worth noting that the use of the algorithm Q (λ) of reverse repetition. This combination shows the fastest convergence

algorithm Q (λ) of reverse repetition of the initial stages of training and high enough value of the average reward at the end of training. If we take into account only those parameters Q (λ), it is much better than other algorithms using reverse repetition.

In summary, I would like to point out that the use of immediate correction of weights provides a better solution and requires fewer resources. The advantage of immediate correction weights also lies in the fact that it can be used to solve problems that are not absorbing state.

REFERENCES

1. **Boyko N. 2016** Basic concepts of dynamic recurrent neural networks development / N. Boyko, P. Pobereyko // ECONTECHMOD : an international quarterly journal on economics of technology and modelling processes. – Lublin: Polish Academy of Sciences. – Vol. 5, № 2. – P. 63-68.
2. **Coelho L. 2013** Building Machine Learning Systems with Python / Luis Pedro Coelho, Willi Richert. – Birmingham – Mumbai: Published by Packt Publishing Ltd. – 290 p.
3. **Bishop C. M. 2006** Pattern recognition and machine learning / Christopher M. Bishop. – Springer Science+Business Media, LLC. – 78 p.
4. **Elkan C. 2003** Using the triangle inequality to accelebrate k-means / C. Elkan // In Proceedings of the Twelfth International Conference on Machine Learning, 2003. – P. 147–153.
5. **Matov O.Ia. 2009** Modern technologies of information resources integration / O.Ia. Matov // Registration, storage and processing of data. - V. 11, № 1, P. 33–42.
6. **Khramova I.O. 2009** The use of service-oriented architectures in the integration of information resources / I.O. Khramova // Registration, storage and processing of data. - V. 11, № 2, P. 70–76.
7. **Matov O.Ia. 2009** Mathematical models of conflict losses performance of the mediators ontology for General use in GRID environment / O.Ia. Matov // Registration, storage and processing of data. - V. 11, № 3, P. 18–25.
8. **Matov O.Ia. 2007** The problem of horizontal integration of information resources in a multi-tiered organizational structures with dynamic configuration / O.Ia. Matov // Registration, storage and processing of data. - V. 9, № 3, P. 88–97.
9. **Matov O.Ia. 2006** Dynamic integration of information resources of the unified information infrastructure of the electricity market / O.Ia. Matov // The functioning and development of electricity and gas markets: collection of scientific works Institute of modelling in energy im. H.Ie. Pukhova. - P. 93–98.
10. **Boyko N. 2016** A look trough methods of intellectual data analysis and their applying in informational systems / N. Boyko // Computer sciences and informatopn technologies CSIT 2016 : Proceedings of XI International scientific conference CSIT 2016 : proceedings. – Lviv: Publ ofv Lviv Polytechnik. – P. 183-185.
11. **Boyko N. 2016** Basic concepts of dynamic recurrent neural networks development / N. Boyko, P. Pobereyko // ECONTECHMOD : an international quarterly journal on economics of technology and modelling processes. – Lublin: Polish Academy of Sciences, Vol. 5, № 2. – P. 63-68.
12. **Leskovec J. 2014** Mining of massive datasets / J. Leskovec, A. Rajaraman, J.D. Ullman. – Massachusetts: Cambridge University Press. – 470 p.
13. **Boyko N. 2017** Use of a cloud storage for implementation informational proce / N. Boyko // ECONTECHMOD : an international quarterly journal on economics of technology and modelling processes – Vol. 2 No.6. – Branch in Lublin: Polish Academy of Sciences, 2017. – P. 3-8.
14. **Boyko N. 2017** Building computer vision systems using machine learning algorithms / N. Boyko, N. Sokil // ECONTECHMOD : an international quarterly journal on economics of technology and modelling processes – Vol. 2 No.6. – Branch in Lublin: Polish Academy of Sciences, 2017. – P. 15-20.
15. **Boyko N. 2016** Using genetic algorithms for modeling informational processes / N. Boyko // Computational problems of electrical engineering : scientific journal "Computational problems elektotehniky" – Vol. 6 No. 1(10) – Founder and Publisher Lviv Polytechnic National University, 2016. – P. 55-62.
16. **Boyko N. 2016** Application of mathematical models for improvement of “cloud” data processes organization / N. Boyko // Mathematical Modeling and Computing : scientific journal "Computational problems elektotehniky" – Vol. 3 No. 2 – Founder and Publisher Lviv Polytechnic National University, 2016. – P. 111-119.
17. **Boyko N.I. 2017** The technological capabilities of the Hyperwave / NI information server. Boyko, O.V. Kopach // International Scientific and Practical Conference "Information Technologies and Computer Modeling", May 15-20, 2017: Abstracts / Repr. for the issue Volodarsky Ye.T. - Ivano-Frankivsk: Mr. Golin O. M. - P. 8-11 p.
18. **Boyko N.I. 2017** Perspective technologies of research of large data in distributed information systems / N.I. Boyko // Radioelectronics, computer science, management. № 4. - Zaporozhye: Zaporizhzhya National Technical University. - P. 66-77.
19. **Maass W. 2002** Real-time computing without stable states: a new framework for neural computations based on perturbations / W. Maass, T. Natschger, H. Markram / Neural Computation : proceedings. – Switzerland: Institute for Theoretical Computer Science, Vol. 11. – P. 2531–2560.
20. **Schrauwen B., Verstraeten D., Campenhout J.V. 2007** An overview of reservoir computing theory, applications and implementations / B. Schrauwen, D. Verstraeten, J.V. Campenhout // Proc. of the 15th European Symp. on Artificial Neural Networks : proceedings. – Belgium: Bruges,. P. 471–482.