e-ISSN 1607-3274 Радіоелектроніка, інформатика, управління. 2021. № 1
p-ISSN 2313-688X Radio Electronics, Computer Science, Control. 2021. № 1

UDC 004.942

DEEP REINFORCEMENT LEARNING WITH SPARSE DISTRIBUTED MEMORY FOR "WATER WORLD" PROBLEM SOLVING

Novotarskyi M. A. – Dr.Sc, Professor of Department of Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.

Stirenko S. G. – Dr.Sc, Professor, Head of Department of Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.

Gordienko Y. G. – Dr.Sc, Professor of Department of Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.

Kuzmych V. A. – Post-graduate student of the Department of Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.

ABSTRACT

Context. Machine learning is one of the actively developing areas of data processing. Reinforcement learning is a class of machine learning methods where the problem involves mapping the sequence of environmental states to agent's actions. Significant progress in this area has been achieved using DQN-algorithms, which became one of the first classes of stable algorithms for learning using deep neural networks. The main disadvantage of this approach is the rapid growth of RAM in real-world tasks. The approach proposed in this paper can partially solve this problem.

Objective. The aim is to develop a method of forming the structure and nature of access to the sparse distributed memory with increased information content to improve reinforcement learning without additional memory.

Method. A method of forming the structure and modification of sparse distributed memory for storing previous transitions of the actor in the form of prototypes is proposed. The method allows increasing the informativeness of the stored data and, as a result, to improve the process of creating a model of the studied process by intensifying the learning of the deep neural network. Increasing the informativeness of the stored data is the result of this sequence of actions. First, we compare the new transition and the last saved transition. To perform this comparison, this method introduces a rate estimate for the distance between transitions. If the distance between the new transition and the last saved transition is smaller than the specified threshold, the new transition is written in place of the previous one without increasing the amount of memory. Otherwise, we create a new prototype in memory while deleting the prototype that has been stored in memory the longest.

Results. The work of the proposed method was studied during the solution of the popular "Water World" test problem. The results showed a 1.5-times increase in the actor's survival time in a hostile environment. This result was achieved by increasing the informativeness of the stored data without increasing the amount of RAM.

Conclusions. The proposed method of forming and modifying the structure of sparse distributed memory allowed to increase the informativeness of the stored data. As a result of this approach, improved reinforcement learning parameters on the example of the "Water World" problem by increasing the accuracy of the model of the physical process represented by a deep neural network.

KEYWORDS: Deep Reinforcement Learning, DQN-algorithm, Sparse Distributed Memory, "Water World" problem.

ABBREVIATIONS

DQN is a Deep Q Network;
RAM is a random access memory;
SDM is a Sparse Distributed Memory.

NOMENCLATURE

Φ is a reinforcement learning method;
S is a state space;
A is a set of permissible actions;
R is a reward function;
Q is a state-action function;
γ is a discount rate;
B is a logical similarity threshold of prototypes;
D is a minibach for deep neural network training;
t is a step of the algorithm;
s_t is a state of the environment at the arbitrary t;
a_t is an action that the agent implements at t;
r_t is a reward that the agent receives at t;
s_{t+1} is a next state of the environment in accordance with t;
f_t is a logical variable that determines whether the t iteration is the final iteration in the current episode;
π is a policy of the deep reinforcement learning;
K is a maximum number of transitions in the SDM;
k is an index of the transition in the SDM;
d_k is a transition tuple that is a prototype in the SDM;
M is a set of minibach indices;
m is an index of the minibach element;
norm is a parameter that determines the norm of similarity of the previous and next states of the environment;
b is a threshold of similarity of the previous and next states of the environment;
N is a coefficient of similarity of prototypes;
n is a dimension of the state vector;
δ is a threshold of the vector element similarities for the previous and next of the environment states;
i is an index of the vector elements of the environment state;
μ_t^i is a similarity of the i-th elements of the current and previous vectors of the environment states;

© Novotarskyi M. A., Stirenko S. G., Gordienko Y. G., Kuzmych V. A., 2021
DOI 10.15588/1607-3274-2021-1-14

P is a SDM queue;
η is a training step size;
J is number of training sessions;
j is a current training session index;
E is a maximum number of episodes;
e is a current episode index.

INTRODUCTION

Reinforcement learning today is a broad class of methods that includes learning how to map the sequence of environmental states to agent's actions. The purpose of the actions of an agent operating in this environment is to maximize the reward it can receive. In turn, the actions of the agent can affect the environment. Therefore, the reinforcement learning methods implement a closed cycle, which significantly distinguishes them from supervised methods of machine learning. Other important features of reinforcement learning are the inability to predict the consequences of the agent's actions accurately and the need to take into account previous actions to increase future rewards. Based on these approaches, a large set of methods has been formed that use linear functions to map the space of the states of the environment to the space of actions of the agent. Linear methods have certain shortcomings, which reduce their efficiency in real-world tasks. One of the most prominent shortcomings is a significant increase in the number of elements of the state space in such tasks. As a result, in most cases, the agent is in a situation for which it has no experience of correct behavior. The only way to solve this problem is to generalize the prior experience gained by the agent and extrapolate it to future states. Algorithms based on deep learning are one of the modern approaches to generalization and have opened a new stage in development of reinforcement learning algorithms. For a long time, these algorithms were considered unstable. The DQN-algorithm has become an important step forward. It was created and tested on the "Arcade" game platform, where DQN showed high training stability. The basic idea behind modern DQN-algorithms is that the global reinforcement learning task is divided into a sequence of local supervised learning tasks. It is implemented by combining the concept of the target network with the experience replay. The data is stored in memory, which is represented by a FIFO-structure of fixed length. The deep learning network uses a training set randomly selected from memory. Ways to improve these methods are mainly based on modifications of data structures that provide the experience replay.

This paper also focuses on improving the data structures of the DQN-algorithm by applying to it the data storage principles used by SDM. This approach aims to reduce the amount of memory used and increase the stability training.

The object of study is the reinforcement learning process of a deep neural network with sparse distributed memory in solving the problem of "Water World".

The subject of the study is a method of improving the deep reinforcement learning without significantly increasing the size of RAM.

The purpose of the work is to develop a method of forming the structure and nature of access to the sparse distributed memory with increased information content to improve reinforcement learning without additional memory.

1 PROBLEM STATEMENT

Reinforcement learning method in general can be represented by a tuple,

$$\Phi = (S, A, Q, R, \gamma),$$

where $Q: S \times A \to A$, $R: S \times A \times S \to \mathbb{R}$, $\gamma \in (0,1)$.

Fig. 1 shows a generalized diagram with the main connections for interaction between the elements of the Φ tuple.

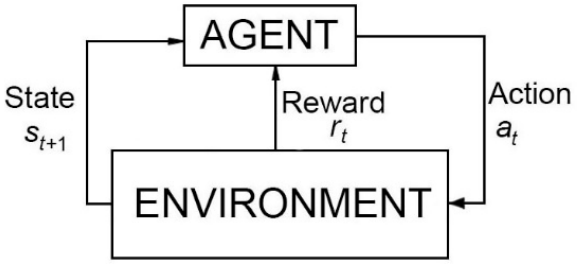

Figure 1 – The generalized scheme of interaction in reinforcement learning

The agent obtains $s_t \in S$ from the environment at an arbitrary t. As a result, the agent implements the $a_t \in A$ and receives a $r_t \in R$ after performing the a_t. The agent also perceives $s_{t+1} \in S$. In this state the environment will be at a $t+1$ time. The purpose of the training system for strengthening is to maximize the total reward, which can be represented as a mathematical expectation of the sum of all rewards with the appropriate discount rates:

$$\mathbf{R} = \mathrm{E}\left[\sum_{t=1}^{T} \gamma^t r_t\right].$$

To do this, in the case of using a DQN-agent, it is necessary to minimize the loss function, the expression of which depends on the chosen method of training.

2 REVIEW OF THE LITERATURE

Unlike other areas of machine learning, in the field of reinforcement learning there is a textbook [1], which most experts consider a basic textbook. This is very important for the unification of terminology and systematization of approaches to the development of new methods. The authors regularly update the content of this textbook. Because of this, it remains relevant for many years. Well-known machine learning algorithms use such mathematical approaches as Finite Markov Decision processes [2],

© Novotarskyi M. A., Stirenko S. G., Gordienko Y. G., Kuzmych V. A., 2021
DOI 10.15588/1607-3274-2021-1-14

Multi-arm bandits [3], gradient descent methods [4], Monte Carlo methods [5], Temporal-Difference Learning [6, 7] and others. When implementing these methods, the exploration-exploitation dilemma always remains relevant [8, 9]. The most commonly used research strategies are: greedy approach, ε -greedy approach, softmax approach and Bayesian Approach [10]. All of these methods have one thing in common, as they require a large amount of memory to store a tabular representation of a value function. Approximate value functions were originally used [11] to overcome this deficiency. Modern approaches to memory reduction involve using SDM [12, 13].

One of the first stable algorithms that uses deep learning for nonlinear approximation of a value function is presented in [14]. This approach is designed to further enhance the capabilities of reinforcement learning [15]. The main advantage of this approach is that it avoided the explicit mapping of the state space to the action space. Combining the benefits of deep learning with the already known machine learning methods, DeepMind has made significant progress in training agent for the game AlphaGo [16]. The created technologies are widely used now for reinforcement learning. This paper uses a modern approach called DQN [17]. Experience replay and target network were used to increase the stability and productivity of learning on small data sets [18]. Features of the application of these technologies for the "Water World" task will be discussed below.

3 MATERIALS AND METHODS

Fig. 2 shows a framework of the deep reinforcement learning system that uses SDM.

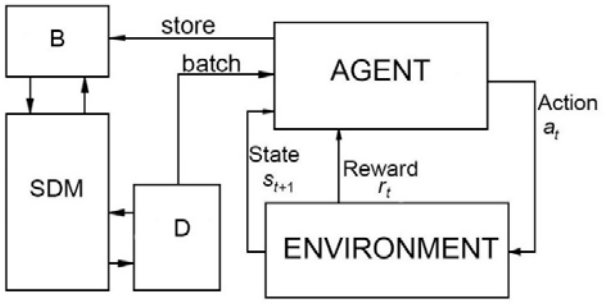

Figure 2 – Deep reinforcement learning system with SDM

As can be seen from Fig. 2, the agent affects the environment by performing the next action. The environment returns a reward for the action performed and notifies its next state, which helps the agent select the next action. This classic scheme of agent-environment interaction is complemented by the fact that each transition is stored by the agent in external memory to implement Experience Replay technology. Information about the transition is represented as the $d_k = (s_t, s_{t+1}, a_t, r_t, f_t)$ tuple, where $0 \le k \le K-1$.

A feature of the new approach we are proposing is the method of storing and reading transitions from SDM. The learning parameters are improved due to the selective storage of only "important" transitions.

Experience Replay technology often uses queued memory. This means that each new transition is always saved at the end of the queue. As soon as the specified amount of memory is full, each addition of a new transition displaces the oldest transition from SDM.

We offer a modified algorithm for storing transitions in SDM. The first step of the algorithm is to compare the new transition with the last transition in the queue. This comparison is as follows:

$$B = norm \;\&\; [a_t = a_{t-1}] \;\&\; [r_t = r_{t-1}] \;\&\; [f_t = f_{t-1}], \quad (1)$$

It should be noted that in formula (1) the expression [.] always returns a certain logical value. The *norm* parameter in expression (1) is determined from formula (2)

$$norm = \left[\min\left(\|s_t - s_{t-1}\|, \|s_{t+1} - s_t\| \right) \ge b \right]. \quad (2)$$

The b value is determined from the expression

$$b = \begin{cases} 1 - \dfrac{1}{N-1}, & N \ge 3, \\ 0.5, & N < 3. \end{cases} \quad (3)$$

The smaller the value of N is, the more states are skipped from storing to SDM due to their similarity to the previously stored states. The use of expression (3) aims to simplify the tuning process.

In the next step of the algorithm, we use the value of B variable to select the method of data storage in the SDM. If the B variable is true, then new transition is written to the place of the last transition in the queue, which corresponds to the time reference $(t-1)$. However, if the B variable is false, then a new transition is added to the end of the queue, and all other transitions are moved in the queue to one position.

Below we define the rule for determining the difference norm for the vectors of states $\|s_t - s_{t-1}\|$ and $\|s_{t+1} - s_t\|$. Let the state vector at time t be represented by elements $s_t = \left(s_t^0, s_t^1, ..., s_t^i, ..., s_t^n\right)$. To calculate the difference norm, first determine the element-by-element similarity of states using the expression:

$$\mu_t^i = \begin{cases} 1 - \dfrac{\left|s_t^i - s_{t-1}^i\right|}{\delta}, & \left|s_t^i - s_{t-1}^i\right| \le \delta, \\ 0, & \left|s_t^i - s_{t-1}^i\right| > \delta, \end{cases} \quad (4)$$

where δ is in the $[0,1]$ range.

Then the difference norm of states s_t and s_{t-1} is $\|s_t - s_{t-1}\| = \min\left(\mu_t^0, \mu_t^1, ..., \mu_t^i, ..., \mu_t^n\right)$. Similarly, using ex-

© Novotarskyi M. A., Stirenko S. G., Gordienko Y. G., Kuzmych V. A., 2021
DOI 10.15588/1607-3274-2021-1-14

pression (4), define the difference norm of s_{t+1} and s_t states as $\|s_{t+1} - s_t\| = \min\left(\mu_{t+1}^0, \mu_{t+1}^1, ..., \mu_{t+1}^i, ..., \mu_{t+1}^n\right)$.

Therefore, a new transition is saved as a new SDM prototype only if the distance between the respective states exceeds the b value or there is a difference between rewards or actions at times t and $(t-1)$. The new prototype is also saved if the new state is the last state in the episode, which is determined by the f_t parameter.

Note, that the optimal definition of difference norm of states is, generally, problem-specific. We show in Section 4 that the proposed difference norm works well for the "Water World" problem.

The memory structure formed by the described algorithm evolves from episode to episode and is used by the learning agent. The agent is an off-policy agent and uses SDM to create a D training set. In the t training step the agent uses a $D_t = \{d_m\}_{m \in M}$ set of transitions. It is obvious that all indices of the selected elements in the M set are in the range $[0, K-1]$.

Algorithm 1	DQN with SDM
1	**Initialization:** P, η, K, J, γ, β, E, T, M
2	**for** $e = 1$ **to** E **do**
	environment.reset()
3	$s \leftarrow$ observe(environment)
4	**while** $t < T$ **do**
5	$a \leftarrow \pi(s)$
6	$t = t + 1$
7	$(s, r, f) \leftarrow$ environment(a)
8	$s_{next} \leftarrow$ observe(environment)
9	update_prototypes(s, s_{next}, r, a, f)
10	**end while**
11	**for** $j = 1$ **to** J **do**
12	minibatch=random.sample(prototypes,M)
13	**for** m **in** M do
14	$s^{array}.append\left(s^m\right)$
15	$s_{next}^{array}.append\left(s_{next}^m\right)$
16	$a^{array}.append\left(a^m\right)$
17	$r^{array}.append\left(r^m\right)$
18	$e^{array}.append\left(e^m\right)$
19	$\delta_j = r_j + \gamma Q\left(s_j, \arg\max_a Q(s_j, a)\right) - Q\left(s_{j-1}, a_{j-1}\right)$
20	$\theta \leftarrow \theta + w_j \delta_j \nabla_\theta Q\left(s_{j-1}, a_{j-1}\right)$ weight change
21	**end for**
22	$\theta \leftarrow \theta + \eta\Delta$ update weights
23	**end for**

Algorithm 2 for writing to SDM and creating new prototypes.

Algorithm 2	update_prototypes(prototype)
1	$(s_t, s_{t+1}, a_t, r_t, f_t)$ = prototype
2	$(s_t, s_{t-1}, a_{t-1}, r_{t-1}, f_{t-1})$ =queue[last]
3	if $N \geq 3$ then $b = 1 - \frac{1}{N-1}$ else $b = 0.5$
4	for $i = 0$ to n do
5	if $\left\|s_t^i - s_{t-1}^i\right\| \leq \delta$ then $\mu_t^i = 1 - \frac{\left\|s_t^i - s_{t-1}^i\right\|}{\delta}$
6	else $\mu_t^i = 0$
7	if $\left\|s_{t+1}^i - s_t^i\right\| \leq \delta$ then $\mu_{t+1}^i = 1 - \frac{\left\|s_{t+1}^i - s_t^i\right\|}{\delta}$
8	else $\mu_{t+1}^i = 0$
9	$\|s_t - s_{t-1}\| = \min\left(\mu_t^0, \mu_t^1, ..., \mu_t^i, ..., \mu_t^n\right)$
10	$\|s_{t+1} - s_t\| = \min\left(\mu_{t+1}^0, \mu_{t+1}^1, ..., \mu_{t+1}^i, ..., \mu_{t+1}^n\right)$
11	$norm = \left[\min\left(\|s_t - s_{t-1}\|, \|s_{t+1} - s_t\|\right) \geq b\right]$
12	if $norm \,\&\, [a_t = a_{t-1}] \,\&\, [r_t = r_{t-1}] \,\&\, [e_t = e_{t-1}]$ then
13	queue[last]=prototype
14	else
15	queue.append(prototype)

The allowable number of episodes, *E,* limits the operation of algorithm 1. In each episode, the algorithm can perform a maximum of *T* iterations if there is no terminal state, which leads to premature termination of the episode. Iteration involves the actor's interaction with the environment. The choice of the next a_t action is based on the current s_t state of the environment in accordance with the π policy. The environment returns a tuple, which contains its new s_t state, the r_t reward for the action and the e_t sign of the terminal state. This tuple, together with the previous s_{t-1} state and a_{t-1} action forms the next prototype for storage in SDM. The method of updating SDM prototypes is implemented in the *update_prototypes*() function, the pseudocode of which is shown in algorithm 2. The essence of this algorithm corresponds to the previous theoretical description. The new prototype is passed to the function as a parameter. The first stage begins with reading the last saved prototype. When fine-tuning the mode of operation of this algorithm, we provide the choice of the constant *N*, which indirectly determines the distance between two successive changes in the vectors of the states of the environment. These norms are calculated in algorithm 2 in accordance with (3) and (4). The resulting step in the analysis of the similarity of the prototypes is the calculation of the logical expression (1). If this expression returns True, then we assume that the new prototype is close to the previous saved prototype. In this case, we replace the previous prototype with a new prototype without increasing the size of the SDM. If expression (1) returns False, it indicates that the actor's action caused

© Novotarskyi M. A., Stirenko S. G., Gordienko Y. G., Kuzmych V. A., 2021
DOI 10.15588/1607-3274-2021-1-14

significant changes in the environment. Therefore, such a prototype is additionally added to the SDM. If the memory is full, we delete the prototype that has been in the SDM for the longest time, as the one that has the least relevance among all saved prototypes. In each episode, we perform *J* sessions of deep neural network training by sequential modification of its θ weights. For each training session, we form a minibatch by randomly selecting prototypes from SDM with a linear probability distribution. Modification of our model allows to define TD-error and to form the graph of the loss function.

This approach has improved reinforcement learning efficiency by making the prototypes stored in SDM more informative without increasing the size of memory to store new information. The results of the study on the example of the "Water World" problem, which confirm the conclusions, are given below.

4 EXPERIMENTS

Consider the work of the described approach on the example of the problem "Water World", which was first proposed in [20, 21] and is considered a popular problem that allows you to explore the reinforcement learning algorithms. The essence of this problem is that round objects float in a square two-dimensional space. The actor is also among those objects that are hostile to him. The purpose of training an actor is to ensure the maximum possible time of the actor's existence, which can be done by avoiding collisions with enemy objects. In order to avoid such collisions, the actor has sensors that are evenly spaced in a circle. The data set created by the set of sensors forms a vector of states of the environment. The actions of the agent in this environment are reduced to the choice of direction and speed modulus in order to avoid collisions. It is important that the actor not only react to the current situation, but also gain experience that would allow him to accept a short-term deterioration of the current situation in order to avoid a catastrophe in the future. An example is a situation where an actor is surrounded by enemy objects and there is a narrow way out of that environment. In this case, the actor must temporarily move closer to enemy objects in order to escape from the environment. The actions of the actor in a similar situation are shown in Fig. 3.

The study was carried out with variation of the parameters of this task to identify their impact on the learning speed of the actor in the proposed method of increasing the informativeness of SDM. Table 1 summarizes the values or ranges of changes in the parameters of the problem.

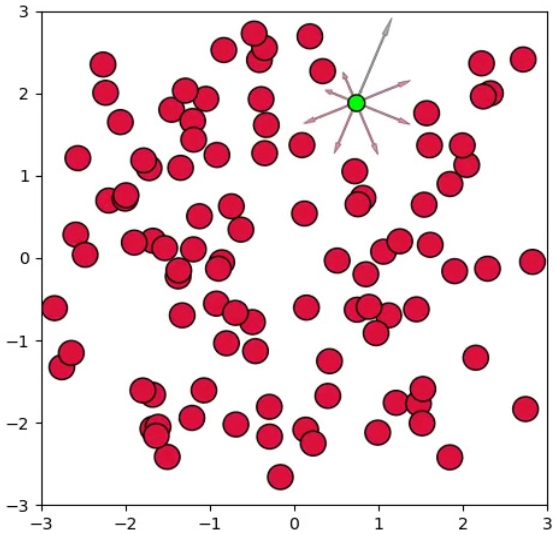

Figure 3 – The actor in a situation of leaving the environment with hostile objects

These studies have confirmed in practice the effectiveness of the proposed approach to increase the informativeness of SDM, as it allowed obtaining improved reinforcement learning when using the same amount of memory. At the same time, when it is necessary to increase the accuracy of learning, the corresponding increase in additional memory is much less than when using the FIFO-structure for memory. The specific values of this reduction require further research.

Fig. 4 shows the number of successful iterations that determine the growth of the "life expectancy" of the actor with the increase in the number of episodes.

On average, with the use of informative SDM, life expectancy increased 1.5 times compared to "Water World", which used a simple queue to store transitions.

Good results can be observed with improved learning parameters due to reduced reinforcement learning error from episode to episode. This fact confirms the graphs of the loss function value decrease (Fig. 5) in comparison with the basic solution of the problem.

Table 1 – Parameters of experiments

№	Parameter name	Range	Unit of measurement
1	Number of sensors	4–64	pcs.
2	Number of enemy objects	20–200	pcs.
3	Number of episodes	200–1000	qty.
4	Number of iterations	2000–4000	qty.
5	SDM size	10 000–100 000	number of prototypes
6	Coefficient of similarity of prototypes, *N*	1–10	qty.
7	Minibatch size	200–400	number of prototypes
8	Survival reward	+1	scores
9	Penalty for collision	–10	scores
10	Discount parameter, γ	0.99	qty.

© Novotarskyi M. A., Stirenko S. G., Gordienko Y. G., Kuzmych V. A., 2021
DOI 10.15588/1607-3274-2021-1-14

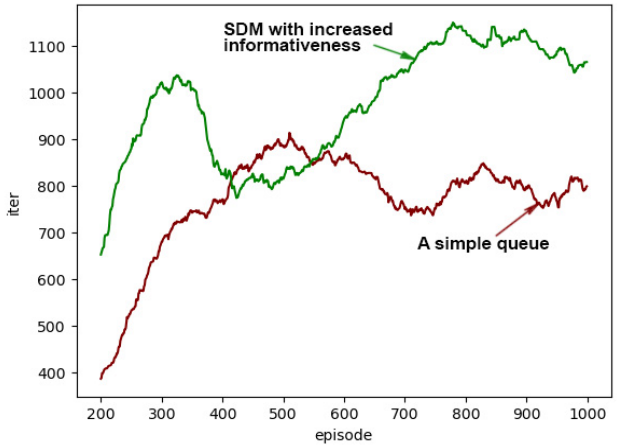

Figure 4 – Dependence of the number of iterations on the number of episodes

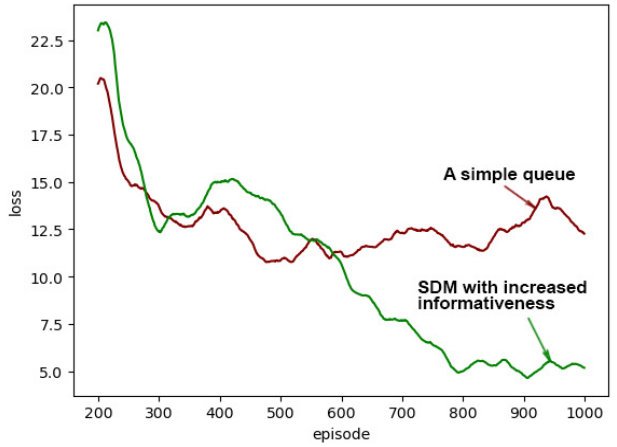

Figure 5 – Comparison of loss functions for the basic approach and method using SDM

In Fig. 6 we see the growth rate of the number of the most successful iterations, that is such iterations in which the actor received a reward of more than 3000 points, provided that the maximum reward for the iteration reaches 4000 points.

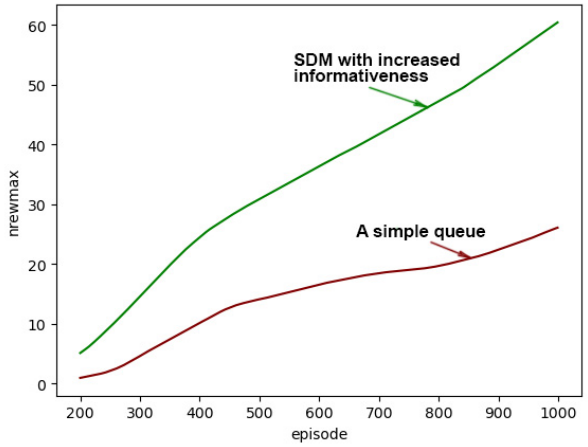

Figure 6 – Comparison of the number of successful iterations for the basic approach and method using SDM

6 DISCUSSION

The paper is devoted to one of the current approaches to reinforcement learning, which uses the deep neural network learning to build a model of the studied process. We used a DQN-algorithm based on two technologies, namely Experience replay and Target network, which allow us to create a sustainable learning process. The peculiarity of this approach is to use a certain buffer with data that reflects the previous experience of the actor, and the use of random samples from this buffer for the deep neural network. The disadvantage of this approach is the need to use large amounts of memory to ensure a successful reinforcement learning process. The main idea of this work is to use SDM to increase the informativeness of the data stored in the buffer. A method of modification and formation of SDM prototypes has also been developed, which provides increased informativeness without increasing its size. Further research, in our opinion, should be aimed at studying the hierarchical structure of SDM, provided that such a hierarchy should be formed on the basis of the hierarchy of features that are elements of the prototypes.

CONCLUSIONS

The article considers a new approach to the creation of DQN – reinforcement learning algorithms. The relevance of these studies is due to the fact that in recent years, algorithms of this type have provided tectonic shifts in the field of learning with reinforcement by allowing the use of neural network neural learning to significantly reduce the amount of RAM used to store previous experience compared to tabular methods reinforced training. Nevertheless, the need for RAM is still significant for real-world problems.

The article proposes an approach that improves the deep reinforcement of learning without a significant increase in memory, which is an urgent problem.

The scientific novelty of the obtained results is that a new method of forming the structure of memory for deep learning is proposed. This method uses a sparse distributed memory structure to store prototypes, each of which is one of the past states of the environment. The main difference of this method from existing analogues is the original principle of adding new prototypes, which allows to increase the informativeness of the stored data. Using this principle, we were able to increase the speed of deep learning by 1.5 times without increasing the size of memory.

The practical significance of the obtained results is that on the basis of this method the software system of training of deep reinforcement is developed. This software system is used to solve an important navigation problem, which is presented in the form of a well-known test task “Water World”. The task is to teach the actor to survive in a dynamically changing environment among hostile objects.

Prospects for further research are to study the hierarchical structure of sparse distributed memory, provided that such a hierarchy should be formed on the basis of a hierarchy of features as elements of prototypes.

© Novotarskyi M. A., Stirenko S. G., Gordienko Y. G., Kuzmych V. A., 2021
DOI 10.15588/1607-3274-2021-1-14

ACKNOWLEDGEMENTS

The work is supported by the state budget scientific research project of National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute". The title of the project is "Artificial Intelligence Platform for Distant Computer-Aided Detection (CADe) and Computer-Aided Diagnosis (CADx) of Human Diseases" (state registration number 0120U105463).

REFERENCES

1. Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. Cambridge: The MIT Press, 2018, 548 p.
2. Puterman M. L. Markov decision processes: discrete stochastic dynamic programming. New Jersey, John Wiley & Sons, 2014, 684 p.
3. Zhao Q. Multi-Armed Bandits: Theory and Applications to Online Learning in Networks. NY, Morgan & Claypool, 2019, 147 p. DOI: 10.2200/S00941ED2V01Y201907CNT022.
4. Theodoridis S. Machine learning: a Bayesian and optimization perspective. Elsevier, 2020, 1160 p. DOI: 10.1016/C2019-0-03772-7.
5. Doucent A., de Freitas N., Gordon N. Sequential Monte Carlo methods in practice. NY, Springer, 2001, 616 p.
6. Hester T. TEXPLORE: Temporal Difference Reinforcement Learning for Robots and Time-Constrained Domains, NY, Springer, 2013, 179 p.
7. Sutton R. Learning to Predict by the Method of Temporal Differences, *Machine Learning*, 1988, Vol. 3, pp. 9–44. DOI: 10.1007/BF00115009.
8. Laureiro-Martinez D., Brusoni S., Canessa N., Zollo M. Understanding the exploration-exploitation dilemma: an fMRI study of attention control and decision-making performance, *Strategic Management Journal*, 2015, Vol. 36, pp. 319–338. DOI: 10.1002/smj.2221.
9. Rejeb L., Guessoum Z., Hallah R. M. An adaptive approach for the exploration-exploitation dilemma for learning agents. Berlin, Springer, 2005, pp. 316–325.
10. Mersmann O., Bischl B., Trautmann H., Preuss M., Weihs C., Rudolph G. Exploratory landscape analysis. *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011, pp. 829–836.
11. Melo F. S., Meyn S. P., Ribeiro M. I. An analysis of reinforcement learning with function approximation, *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 664–671.
12. Kanerva P. Sparse distributed memory. Cambridge:The MIT Press, 1990, 155 p.
13. Wu Ch. Novel Function Approximation Techniques forLarge-scale Reinforcement Learning. PhD dissertation, 2010, 130 p.
14. Mnih V., Kavukcuoglu K., Silver D., Graves A., Antonoglou I., Wierstra D., Riedmiller M. A. Playing Atari with Deep Reinforcement Learning, *arXiv*: 1312.5602v1 [cs. LG], 2013, 9 p.
15. Ollero J., Child C.H.T. Performance Enhancement of Deep Reinforcement Learning Networks using Feature Extraction, *Lecture Notes in Computer Science*, 2018, Vol. 10878. pp. 208–218. DOI: 10.1007/978-3-319-92537-0_25.
16. Holcomb S. D. Porter W.K., Ault Sh. V., Mao G., Wang J. Overview on DeepMind and its AlphaGo Zero AI, *Proceedings of 2018 International Conference on Big Data and Education*, 2018, pp. 67–71. DOI: 10.1145/3206157.3206174.
17. Sewak M. Deep Q Network (DQN), Double DQN, and Dueling DQN, *Deep Reinforcement Learning*. Springer, pp. 95–108. DOI: 10.1007/978-981-13-8285-7_8.
18. Gao J., Shen Y., Liu J., Ito M., Shiratori N. Adaptive traffic signal control : deep reinforcement learning algorithm with experience replay and target network, arXiv 1705.02755v1 [cs.N], 2017, 10 p.

Received 15.12.2020.
Accepted 25.01.2021.

УДК 004.942

ГЛИБОКЕ НАВЧАННЯ З ПІДКРІПЛЕННЯМ З ПАМ'ЯТТЮ З ПРОРІДЖЕНИМИ ДАНИМИ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ «ВОДНИЙ СВІТ»

Новотарський М. А. – д-р техн. наук, професор кафедри обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

Стіренко С. Г. – д-р техн. наук, професор, завідувач кафедри обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

Гордієнко Ю. Г. – д-р техн. наук, професор кафедри обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

Кузьмич В. А. – аспірант кафедри обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна.

АНОТАЦІЯ

Актуальність. Машинне навчання це одна з галузей обробки даних, яка активно розвивається. Значних успіхів у цій сфері вдалося досягти завдяки використанню DQN-алгоритмів, які стали одними з перших стійких алгоритмів навчання при використанні глибоких нейронних мереж. Основним недоліком такого підходу є стрімке зростання оперативної пам'яті при реалізації задач реального світу. Запропонований в роботі підхід дозволяє частково вирішити цю проблему.

Мета. Метою роботи є розробка методу формування структури та характеру доступу до розрідженої розподіленої пам'яті з підвищеною інформативністю для покращення навчання з підкріпленням без залучення додаткової пам'яті.

Метод. Запропоновано метод формування структури та модифікації пам'яті з проріженими даними для зберігання попередніх переходів актора у вигляді прототипів. Метод дозволяє підвищити інформативність збережених даних і, як результат, покращити процес створення моделі досліджуваного процесу шляхом інтенсифікації навчання глибокої нейронної мережі. Підвищення інформативності збережених даних є результатом такої послідовності дій. Спочатку виконуємо порівняння нового переходу та останнього збереженого переходу. Для виконання такого порівняння, в рамках даного методу, введено норму оцінки відстані між переходами. Якщо відстань між новим переходом та останнім збереженим переходом є меншою за заданий поріг, то новий перехід записується на місце попереднього без збільшення обсягу пам'яті. У протилежному випадку створюємо новий прототип в пам'яті з одночасним видаленням прототипу, який зберігався у пам'яті найдовше.

Результати. Роботу запропонованого методу було досліджено під час вирішення популярної тестової задачі "Водний світ". Результати показали збільшення часу виживання актора у ворожому середовищі в 1,5 рази. Такий результат був досягнутий за рахунок підвищення інформативності збережених даних без збільшення обсягу оперативної пам'яті.

Висновки. Запропонований метод формування та модифікації структури пам'яті з проріженими даними дозволив підвищити інформативність збережених даних. В результаті такого підходу було одержано покращені параметри навчання з підкріпленням на прикладі задачі «Водний світ» за рахунок підвищення точності моделі фізичного процесу, представленого глибокою нейронною мережею.

КЛЮЧОВІ СЛОВА: глибоке навчання з підкріпленням, DQN-алгоритм, розріджена розподілена пам'ять, задача «Водний світ».

© Novotarskyi M. A., Stirenko S. G., Gordienko Y. G., Kuzmych V. A., 2021
DOI 10.15588/1607-3274-2021-1-14

УДК 004.942

ГЛУБОКОЕ ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ С ПАМЯТЬЮ С ПРОРЕЖЕННЫМИ ДАННЫМИ ДЛЯ РЕШЕНИЯ ЗАДАЧИ «ВОДНЫЙ МИР»

Новотарский М. А. – д-р техн. наук, профессор кафедры вычислительной техники, Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского», Киев, Украина.

Стиренко С. Г. – д-р техн. наук, профессор, заведующий кафедрой вычислительной техники, Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского», Киев, Украина.

Гордиенко Ю. Г. – д-р техн. наук, профессор кафедры вычислительной техники, Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского», Киев, Украина.

Кузьмич В. А. – аспирант кафедры вычислительной техники, Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского», Киев, Украина.

АННОТАЦИЯ

Актуальность. Машинное обучение это одна из активно развивающихся отраслей обработки данных. Значительных успехов в этой сфере удалось достичь благодаря использованию DQN-алгоритмов, которые стали одними из первых устойчивых алгоритмов обучения с использованием глубоких нейронных сетей. Основным недостатком такого подхода является стремительный рост оперативной памяти при реализации задач реального мира. Предложенный в работе подход позволяет частично решить эту проблему.

Цель. Целью работы является разработка метода формирования структуры и характера доступа к разреженной распределенной памяти с повышенной информативностью для улучшения обучения с подкреплением без привлечения дополнительной памяти.

Метод. Предложен метод формирования структуры и модификации памяти с прореженными данными для хранения предыдущих переходов актера в виде прототипов. Метод позволяет повысить информативность хранимых данных и, как результат, улучшить процесс создания модели изучаемого процесса путем интенсификации обучения глубокой нейронной сети. Повышение информативности хранимых данных является результатом такой последовательности действий. Сначала выполняем сравнение нового перехода и последнего сохраненного перехода. Для выполнения такого сравнения, в рамках данного метода, введена норма оценки расстояния между переходами. Если расстояние между новым переходом и последним сохраненным переходом меньше заданного порога, то новый переход записывается на место предыдущего без увеличения объема памяти. В противном случае создаем новый прототип в памяти с одновременным удалением того прототипа, который хранился в памяти дольше.

Результаты. Работа предложенного метода была исследована в ходе решения популярной тестовой задачи «Водный мир». Результаты показали увеличение времени выживания актера во враждебной среде в 1,5 раза. Такой результат был достигнут за счет повышения информативности хранимых данных без увеличения объема оперативной памяти.

Выводы. Предложенный метод формирования и модификации памяти с прореженными данными позволил повысить информативность хранимых данных. В результате такого подхода были получены улучшенные параметры обучения с подкреплением на примере задачи «Водный мир» за счет повышения точности модели физического процесса, представленного глубокой нейронной сетью.

КЛЮЧЕВЫЕ СЛОВА: глубокое обучение с подкреплением, DQN-алгоритм, разреженная распределенная память, задача «Водный мир».

ЛІТЕРАТУРА / ЛИТЕРАТУРА

1. Sutton R. S. Reinforcement Learning: An Introduction / R. S. Sutton, A. G. Barto. – Cambridge : The MIT Press, 2018. – 548 p.
2. Puterman M.L. Markov decision processes: discrete stochastic dynamic programming / M. L. Puterman. – New Jersey : John Wiley & Sons, 2014. – 684 p.
3. Zhao Q. Multi-Armed Bandits: Theory and Applications to Online Learning in Networks / Q. Zhao. – NY : Morgan & Claypool, 2019. – 147 p. DOI: 10.2200/S00941ED2V01Y201907CNT022.
4. Theodoridis S. Machine learning: a Bayesian and optimization perspective / S. Theodoridis. – Elsevier, 2020. – 1160 p. DOI: 10.1016/C2019-0-03772-7.
5. Doucent A. Sequential Monte Carlo methods in practice / A. Doucent, N. de Freitas, N. Gordon. – NY : Springer, 2001. – 616 p.
6. Hester T. TEXPLORE: Temporal Difference Reinforcement Learning for Robots and Time-Constrained Domains / T. Hester. – NY : Springer, 2013. – 179 p.
7. Sutton R. Learning to Predict by the Method of Temporal Differences / R. Sutton // Machine Learning. – 1988. – Vol. 3. – P. 9–44. DOI: 10.1007/BF00115009.
8. Understanding the exploration-exploitation dilemma: an fMRI study of attention control and decision-making performance / [D. Laureiro-Martinez, S. Brusoni, N. Canessa, M. Zollo] // Strategic Management Journal. – 2015. – Vol. 36. – P. 319–338. DOI: 10.1002/smj.2221.
9. Rejeb L. An adaptive approach for the exploration-exploitation dilemma for learning agents / L. Rejeb, Z. Guessoum, R. M'Hallah. – Berlin : Springer, 2005. – P. 316–325.
10. Mersmann O. Exploratory landscape analysis / [O. Mersmann, B. Bischl, H. Trautmann et al] // Proceedings of the 13th annual conference on Genetic and evolutionary computation. – 2011. – P. 829–836.
11. Melo F. S. An analysis of reinforcement learning with function approximation / F. S. Melo, S. P. Meyn, M. I. Ribeiro // Proceedings of the 25th international conference on Machine learning. – 2008. – P. 664–671.
12. Kanerva P. Sparse distributed memory / P. Kanerva. – Cambridge : The MIT Press, 1990. – 155 p.
13. Wu Ch. Novel Function Approximation Techniques forLarge-scale Reinforcement Learning / Ch. Wu. – PhD dissertation, 2010. – 130 p.
14. Playing Atari with Deep Reinforcement Learning / [V. Mnih, K. Kavukcuoglu, D. Silver et al] // arXiv: 1312.5602v1 [cs. LG], 2013. – 9 p.
15. Ollero J. Performance Enhancement of Deep Reinforcement Learning Networks using Feature Extraction / J. Ollero, C.H.T. Child // Lecture Notes in Computer Science. – 2018. – Vol. 10878. – P. 208–218. DOI: 10.1007/978-3-319-92537-0_25
16. Overview on DeepMind and its AlphaGo Zero AI / [S. D. Holcomb, W. K. Porter, Sh. V. Ault et al] // Proceedings of 2018 International Conference on Big Data and Education. – 2018. – P. 67–71. DOI: 10.1145/3206157.3206174.
17. Sewak M. Deep Q Network (DQN), Double DQN, and Dueling DQN / M. Sewak // Deep Reinforcement Learning. – Springer – P. 95–108. DOI: 10.1007/978-981-13-8285-7_8.
18. Gao J. Adaptive traffic signal control : deep reinforcement learning algorithm with experience replay and target network / [J. Gao, Y. Shen, J. Liu et al] // arXiv 1705.02755v1 [cs.N] . – 2017. – 10 p.

© Novotarskyi M. A., Stirenko S. G., Gordienko Y. G., Kuzmych V. A., 2021
DOI 10.15588/1607-3274-2021-1-14