Applying a neural network architecture with spatio-temporal connections to the maze exploration

Dmitry Filin and Aleksandr I. Panov

Abstract We present a model of Reinforcement Learning, which consists of modified neural-network architecture with spatio-temporal connections, known as Temporal Hebbian Self-Organizing Map (THSOM). A number of experiments were conducted to test the model on the maze solving problem. The algorithm demonstrates sustainable learning, building a near to optimal routes. This work describes an agents behavior in the mazes of different complexity and also influence of models parameters at the length of formed paths.

1 Introduction

Currently, the problem of increasing a level of robotic systems autonomy due to integration extended knowledge and learning subsystems into their control systems becomes an important direction in artificial intelligence and cognitive architectures [6, 7, 8]. Deep Learning with Reinforcement demonstrated impressive results on the so-called ”raw data”, i.e. unprocessed images obtained from sensors of a learning system [1]. At present, systems developed for simple experiments in the game simulation environments are beginning to be used in real robotic tasks [3, 4]. The idea of using an information received from such sensors as visual and sound as a training data, makes it possible to construct a representation of the environment which an agent interacts with. Integration of neural networks with traditional Q-learning

Dmitry Filin
National Research University Higher School of Economics, Moscow, Russia
e-mail: dafilin@edu.hse.ru

Aleksandr I. Panov
Federal Research Center “Computer Science and Control” of Russian Academy of Sciences, Moscow, Russia
National Research University Higher School of Economics, Moscow, Russia
e-mail: apanov@hse.ru

makes it possible to match observed states of environment with reward received from it for certain actions. An automatic generation of characteristics for a better description of environment states using neural networks makes it possible to apply Q-learning to real "raw data" obtained from sensors [5].

In this paper, we present a model combining one of the neural network architectures based on THSOM [2], with Q-learning for the maze solving. Our motivation is the idea that, in general, all the labyrinths (their images), consist of some patterns (see Fig. 1), which can be divided into several categories. If an agent understood in what state of the environment it is, it could determine the most appropriate action for this state. So, for example, in the case shown in Fig. 2, the agents available actions are "move left" and "move down".

Initially, the THSOM algorithm is used to generate Markov sequences (namely, for recovering the probabilities of transitions between system states) according to the input data stream. Therefore, it seems intuitively that such an approach could be successfully integrated into the concept of Q-learning. Due to the clustering of the input data, we additionally get a reduction in the dimensionality of the environment states space with the minimal loss of information. In contrast to the traditional approach [9], where the Q-table stores all environment states, we assert that only a few states are enough to describe a full motion process of an agent. Using the Q-network architecture developed in DeepMind [1], we denoted the agent's observed state of the environment, as an image of a maze section from the top view. While the agent is moving, it "sees" the $M \times M$ field around itself. It should be noted that initially the agent does not know anything about the environment where he moves. The process of clustering is taking place during the process of an environment exploration by the agent, thus independence from the map of the labyrinth is achieved. Finally, the main difference from the classical Q-learning is that all information about movement is stored within neuronal connections without the use of additional Q-tables, in other words, the connections between sensory and motor parts of the cerebral cortex are modeled [10].

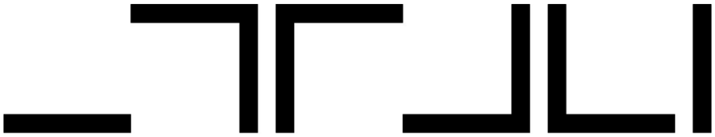


Fig. 1 Examples of maze patterns

2 Formal problem

Consider a stochastic environment in which an agent can access the following set of actions: $A = [L, R, U, D]$. Corresponding spatial increments are $\Delta = [(0,-1); (0,1); (-1,0); (1,0)]$. Each maze cell is in one of the following states: $S = [0, \#, F]$. The first is for free cell,

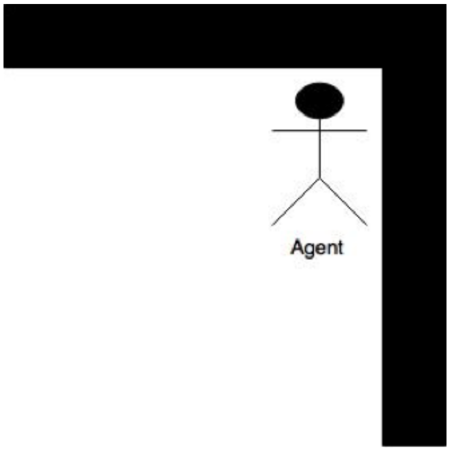
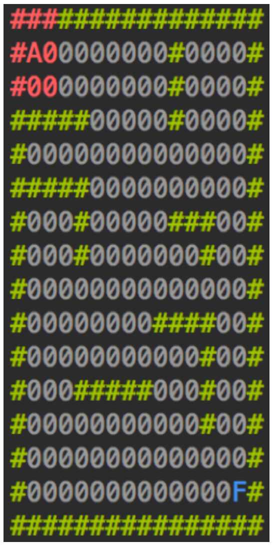


Fig. 2 An agent observes a specific environment state

the second is for the wall and the last denotes finish point. When the agent makes a step it receives a reward: $R = [SR, WR, FR]$.

Additionally we give the agent a reward for the finish point approach, we describe it in the next section. Our target is to achieve an aim with the minimum number of steps. An example of an emulated environment is attached below.



```
################
#A00000000#0000#
#000000000#0000#
#####00000#0000#
#00000000000000#
#####0000000000#
#000#00000###00#
#000#0000000#00#
#00000000000000#
#00000000####00#
#00000000000#00#
#000#####000#00#
#00000000000#00#
#00000000000000#
#0000000000000F#
################
```

Fig. 3 An emulated maze. The red area is corresponding to the agent's observation

3 Neural network architecture

As already mentioned, the model is based on the neural network THSOM (Fig. 4), which has two types of inter-neural connections - spatial and temporal. The basic idea is that each input vector is an attractor for neurons, to which some of them are iteratively converge in the learning process, thus clusters of similar patterns are formed. This process is controlled by the standard for learning without a teacher method (for example, Kohonen self-organizing map algorithm). The BMU radius is calculated using the following equation:

$$r = r_0 * \exp\left(\frac{-t}{r_1}\right) \tag{1}$$

but in our experiment it is zero so that only one neuron converges. In this way, the smallest number of necessary states is required, and there is no significant effect on the efficiency of the algorithm. The strength of neural connections adjusted according to the following equation:

$$\bar{x} + = s_0 \exp\left(\frac{dist^2}{s_1}\right) * t_0 \exp\left(\frac{-t}{t_1}\right) * (\bar{y} - \bar{x}) \tag{2}$$

where $\bar{x}$ represents a neuron, $\bar{y}$ is an input vector, $dist$ - a distance between the vectors.

The calculation of distance deserves special attention here, since in our problem it is a measure of similarity of the labyrinth patterns. Standard metrics are not suitable, since they are based on the difference of the corresponding components of the vector, whereas in the case below, the patterns (see Fig. 5) can be considered the same, as they correspond to the same available agent actions. However, in Fig. 6 patterns though similar in appearance, but differ in the set of actions available to the agent. That means, if position of the wall relative to the agent is not important for him, and he cares only about the form, then, having learned to walk upwards from a horizontal obstacle, he will do this always and there will be no difference from which side of the wall he is. In this regard, a metric was introduced that takes into account both the difference between the structure of the patterns and their location in the area of visibility. Then the distance is considered as:

$$dist(P_1, P_2) = \alpha * shift + (1 - \alpha) * diff \tag{3}$$

where $diff$ is the minimum difference between patterns when overlapping them (in units), $shift$ the number of shifts to achieve this difference.

The temporal component of the model consists of 4 connections between each pair of neurons that determine the probability distribution in the action space. That is, the stronger the connection, the more likely the action to be taken to move from one state to another. The calculation of temporary weights is according to the formula:

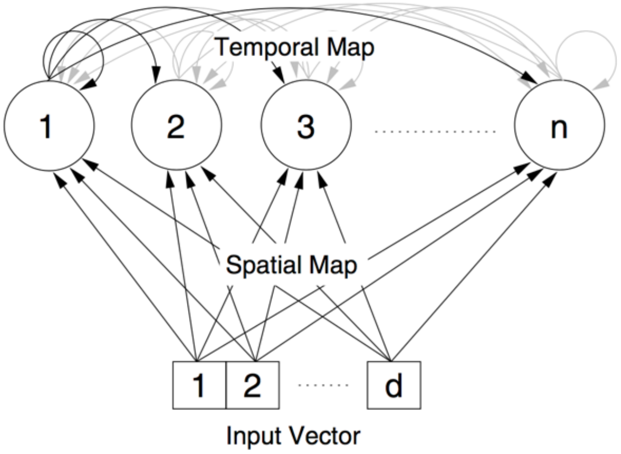


Fig. 4 THSOM Architecture

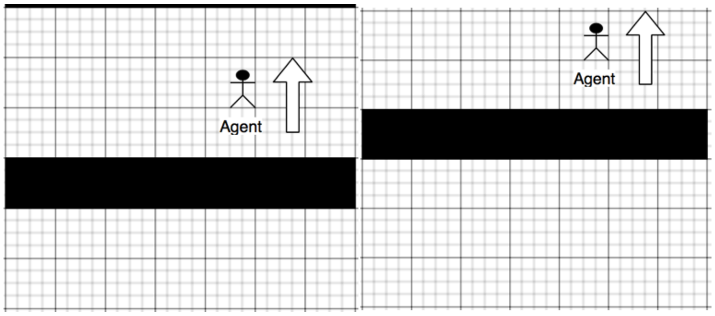


Fig. 5 An example of similar patterns

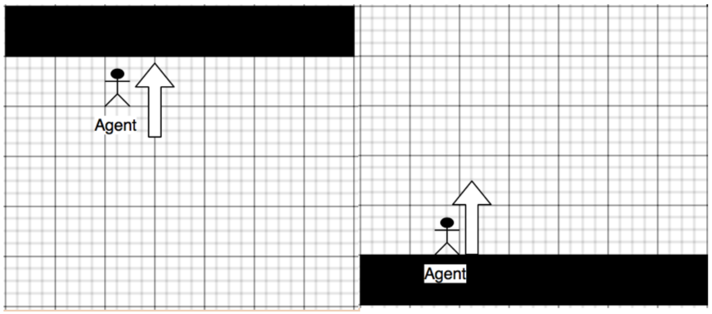


Fig. 6 Patterns have the same structure, but in the first case upward action is impossible

$$w^t_{i,j,a} = \min(\max(w^{t-1}_{i,j,a} + reward, 0), 1) \tag{4}$$

where i and j are environment states, a is an action, *reward* is an environment feedback. It consists of a constant reward / punishment for each step + reward for approaching to the final point. The last component was added to ensure that the agent is motivated to go exactly to the finish. The choice of an action at the moment of time t corresponds to the strongest outgoing link from the current active neuron.

In addition to choosing the optimal action, the model also uses a strategy that plays an extremely important role in the initial stages. When the process is just beginning and the agent does not know anything about the patterns, it acts for a while absolutely randomly, accumulating experience. In the general case, the probability of a random action is calculated as:

$$P_\varepsilon = \max\left(\exp\left(\frac{-t}{M_1}\right), 0.2\right) \tag{5}$$

3.1 Algorithm

```
1. Initialize time and space weights with small random values
2. For t=1, T do
    2.1. Get an input signal from the sensors (get the current MxM block)
    2.2. Find BMU
    2.3. Update spatial weights
    2.4. if t != 1 then update temporal weights
    2.5. Remember current state as $prev\_state$
    2.6. According to $\varepsilon - greedy$ policy choose the $best\_action$
    2.7. Do the chosen action, remember $reward$
    2.8. If finish point is not achieved yet goto step 2
3. end for
```

4 Experiments

We tested our algorithm on various sequences of labyrinths 16x16, differing in complexity of structure. To begin with, it was decided to run the model 5 times on the same labyrinth (Fig. 3) to understand how well the learning goes. The number of steps is indicated in the following table:

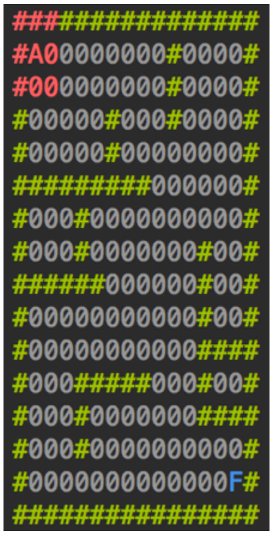
Table 1

Iteration	1	2	3	4	5
Steps	119	84	49	33	42

As can be seen, the agent is learning and learning quite quickly. Taking into account the initial location of the agent and the end point location, the results can be considered comparable to human. However, in more complex labyrinths, the agent needs much more steps. For example, if the labyrinth contains dead ends, the agent is spending extra time in order to get out of it, which affects the total time. In this case, as in Fig. 7, the statistics are as follows:

Table 2

Maze	1	2	3	4	5
Steps	784	702	522	284	238



```
################
#A00000000#0000#
#000000000#0000#
#00000#000#0000#
#00000#00000000#
#########000000#
#000#0000000000#
#000#0000000#00#
######000000#00#
#00000000000#00#
#00000000000####
#000#####000#00#
#000#0000000####
#000#0000000000#
#0000000000000F#
################
```

Fig. 7 Complex maze

Next, we tested the algorithm on a sequence of five different labyrinths several times and averaged the results:

Table 3

Iteration	1	2	3	4	5
Steps	529	758	329	557	205

It should be noted that all labyrinths were different relative to each other in terms of the complexity of the patterns and the location of the start and end points. In the general case, during the experiments, the following circumstances that influenced the operation time of the algorithm were elucidated:

1. Constants that control the speed of learning, clustering, as well as the number of neurons and the like. For example, the lower the intensity of clustering, what means, the lower the strength of spatial connections, the longer, but more qualitatively, the algorithm works. Low weights should be considered in the case of labyrinths with a complex structure of patterns, where individual cells affect the movements of the agent (for example, narrow passages, non-standard wall bends).
2. The location of the start and end points. Since the agent is given a reward for approaching the finish line, he has his own priorities in the movements when passing obstacles. Therefore, if the agent is trained first on a sample of labyrinths, where the finish is at the bottom right, and then given the opportunity to walk through the labyrinth, the end point in which is located on the left from above, this can significantly affect the operation time. However, even in this case, the agent will pass such a labyrinth faster than if he had seen it for the first time.
3. The initial location of the agent on the map. The extent to which the experience of the agent during the research will be varied, following the $\varepsilon - greedy$ rule. The more diverse obstacles that the agent will encounter during this period, what means, the more he learns, the more accurately he will move, based solely on his own experience.

5 Conclusion

In this paper, we present an original neural network architecture of an intelligent agent capable of learning how to build paths in various labyrinths. The architecture is based on the well-known THSOM model, with modifications for use in the learning with reinforcement problems. Based on the results of the conducted experiments, we can conclude that the learning process converges. In general, the agent not only always finds a way, but does it reasonably quickly. In the future, it is planned to more thoroughly analyze each of the model parameters in order to achieve the best time result. Also we are working on a method that will help the agent to deal with complicated situations as dead ends and fake ways and not get stuck in them for a long time.

Acknowledgements The reported study was supported by RFBR, research Projects No. 16-37-60055 and No. 15-07-06214.

References

1. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller: ”Playing Atari with Deep Reinforcement Learning”, 2013.
2. Koutnk and M. norek: “Temporal Hebbian Self-Organizing Map for Sequences”, 2008
3. Gupta S. et al. Cognitive Mapping and Planning for Visual Navigation // ArXiv: 1702.03920.

4. Schrodt F. et al. Mario Becomes Cognitive // Top. Cogn. Sci. 2017. P. 131.
5. Paxton C. et al. Combining Neural Networks and Tree Search for Task and Motion Planning in Challenging Environments // ArXiv: 1703.07887. 2017.
6. Broy, M.: Software engineering — from auxiliary to key technologies. In: Broy, M., Dener, E. (eds.) Software Pioneers, pp. 10-13. Springer, Heidelberg (2002)
7. Panov A.I. Behavior Planning of Intelligent Agent with Sign World Model // Biol. Inspired Cogn. Archit. 2017. Vol. 19. P. 2131.
8. Emelyanov S. et al. Multilayer cognitive architecture for UAV control // Cogn. Syst. Res. 2016. Vol. 39. P. 5872.
9. Kaelbling L.P., Littman M.L., Moore A.W. Reinforcement learning: A survey // J. Artif. Intell. Res. 1996. Vol. 4. P. 237285.
10. Chalita M.A., Lis D., Caverzasi A. Reinforcement learning in a bio-connectionist model based in the thalamo-cortical neural circuit // Biol. Inspired Cogn. Archit. 2016. P. 4563.