© Journal of the Technical University - Sofia
Plovdiv branch, Bulgaria
"Fundamental Sciences and Applications"
Vol. 25, 2019

Applying Reinforcement learning to find the logic puzzles solution

Dilyana Budakova, Velyo Vasilev

Technical University of Sofia, Branch Plovdiv, 61, "Sankt Peterburg" blvd. Bulgaria

dilyana_budakova@yahoo.com, velyo.vasilev@abv.bg

Abstract. Solving problems, which are presented as games can help find solutions to complex tasks, which are divided into subtasks. In this article Reinforcement learning is applied in order to find the solutions to some logic games. Experiments on the training of the modeled agent with different values of the learning parameter have been performed and conclusions have been drawn.

1. Introduction

Reinforcement learning is a type of learning, in which a trainee learns what to do, namely how to make a match between a situation and an action, so as to maximize a numerical reward received as a signal. The trainee has to discover which actions result in the highest reward and attempt to employ them. When choosing an action, not only the immediate reward is taken into account, but also the next situations and the future rewards [1].

The one being trained is the agent. The agent is implemented through a program, which implements the function of the agent. The function of the agent makes a match between what the agent perceives and the actions, which it takes. All Reinforcement learning agents have explicit goals, they can detect aspects of their environment and can choose actions, which influence it [1].

In the real world reinforcement learning [5] is used for Traffic Light Control [6], in robotics [9], for optimizing chemical reactions [10], Display Advertising [7], to solve different games [11].

In the field of artificial intelligence, classical puzzles, games and problems are often used as toy problems [2,8]. These include sliding-block puzzles, N-Queens problem, missionaries and cannibals problem, tick-tack-toe, chess, Tower of Hanoi and others. In this article the Reinforcement learning algorithm is applied in order to find a solution to several games and problems, some of which are: Tower of Hanoi, Solving a maze [4], Farmer crosses river puzzle, Missionaries and cannibals problem [3].

2. Realizing the training of the agent

In order to carry out the training process, the learning agent is given: The environment model (fig.1); The rewards model; the behaviour function of the agent; A value of the learning parameter.

2.1. Environment model

The environment model represents a graph of the environment states. For example when modelling the problem of finding a solution to the Missionaries and cannibals puzzle, the vertices of the graph are the possible states and the edges show which states can be transitioned between (fig. 1).

2.2. Rewards model
The rewards model is necessary to give the agent a goal. For example when the goal is for all objects to cross a river from the east coast to the west coast, then the agent will receive a reward only when all the objects reach the west coast.

2.3. The behaviour function of the agent
The current state is evaluated when choosing a given action, in other words the benefit of taking a certain action, given a certain state is calculated. To accomplish this the following needs to be taken into account 1) the immediate reward, which the agent receives when choosing the action in question, given a certain situation *R(state, action)*; 2) the evaluation of the best action, which the agent can take in the next state, in which it will arrive if it takes the action in question, given the situation: *Max[Q(next state, all actions)]; 3)* the value of the learning parameter *ϒ ,* which is in the interval between 0 and 1 ($0 \leq ϒ < 1$). If ϒ is closer to zero, the agent would prefer to only go after the immediate reward. If ϒ is closer to one, then the agent will put higher priority on the future reward.

The agent training formula is as follows:

Q(state,action)=R(state,action)+ϒ.Max[Q(next_state,all_actions)]

2.4. Modelling the memory of the agent
The memory of the agent is represented by a Q matrix. The rows of the Q matrix represent the current state of the agent, and the columns are the states, in which the agent can go.

At the start it is assumed, that the agent has no knowledge, so all elements of the Q matrix are equal to zero. The values of the Q matrix are changed by applying the virtual agent training formula.

3. Realized reinforcement learning algorithm steps

1. The learning parameter ϒ is assigned a value and a matrix with the rewards, which will be received from the environment R, is set.
2. The initial values of the Q matrix (the memory model of the agent) are set to zero
3. For every episode:

```
    - A random initial state is selected
    - The following steps are repeated until the goal is reached:
            One of the multiple possible actions from the current state is selected.
            The state in which the agent will go, based on the selected action, is examined. All possible
        actions from this next state are examined. The highest action value for this next state is taken.
            > The current state is evaluated based on this choice of action. The immediate reward is
        taken into account as well as the highest evaluated action which can be taken in the next state
        where the agent will arrive with this choice of action:
            > The next state is set as current
    - End of the loop
```

4. End of the episode

4. Functions of the program realization

The used development environment is Visual Studio.NET and the programming language is C#.

The following, more important, functions have been realized:

find_naslednitzi - a function to find all possible actions for a given state

biggest_naslednik - a function to find the action with the highest numerical reward(max).

calculate - calculates the aforementioned virtual agent training formula Q(state, action) = R(state, action) + ϒ. Max[Q(next state, all actions)].

learning - fills the Q matrix – the memory of the agent.

testing - derives the optimal solution to a given game.

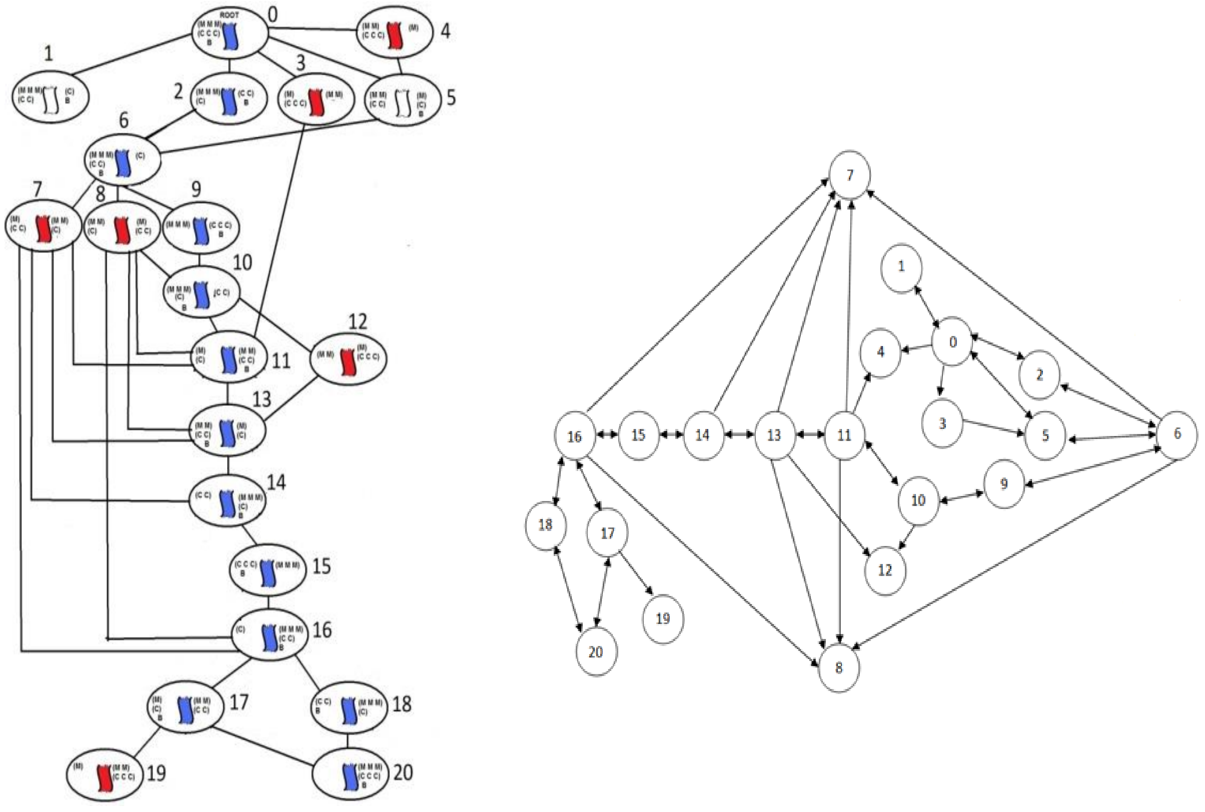

Fig. 1. Model of the environment of the Missionaries and cannibals problem.

5. Experiment and experimental results

Experiments on the training of the modelled agent with different values of the learning parameter have been performed and conclusions about the number of necessary training episodes have been drawn. It was experimentally revealed, that when the values of the training parameter are close to 0, the modelled agent cannot be trained. This is the case, because in this scenario the immediate reward is primarily taken into account. In this case the evaluation of the states does not change, regardless of the number of episodes.

Therefore for a successful training it is required to not only take into account the immediate reward, but also the future reward. That is why the value of the learning parameter has to be close to 1. It was experimentally revealed, that the quickest and most successful training happens when the learning parameter is $\Upsilon = 0.8$. This is a value close to 1, at which the future reward plays a bigger role in the training process. At values of the training parameter in the order of $\Upsilon = 0.85$; $\Upsilon = 0.80$; or $\Upsilon = 0.70$ the training is successful even after a number of episodes in the order of 100 - 200. The maximal number of episodes, at a learning parameter of 0.8, for each of the implemented games, is given on fig. 2. After this number of episodes the evaluations of the states do not change. That is to say the training does not go on for an infinite amount of time.

Name of the game	Number of episodes
Tower of Hanoi	600
Maze	800
Farmer crosses river	400
Missionaries and cannibals	900

Fig. 2 Maximal number of episodes when training the modelled agent.

6. Conclusion

In this article the implementation of a Reinforcement learning algorithm and its application for finding the solution to logic games are examined. Experiments about the training of the modelled agent at different values of the learning parameter have been performed.

It was experimentally revealed, that the successful training requires taking into account not only the immediate reward, but also the future reward. This is achieved through setting a value of the learning parameter close to one. It was experimentally revealed, that the training is fastest when the learning parameter is $\Upsilon = 0.8$. In order for the algorithm to work the following has been created for each game: Environment model; Rewards model; The behaviour function of the agent; the learning parameter is given a value.

References

[1] Stuart J. Russell, Peter Norvig (2010). Artificial Intelligence: A Modern Approach (3 ed.). pp. 70–73, 102–107, 109–110, 115, 162. ISBN 978-0-13-604259-4.

[2] Toy problem, From Wikipedia, the free encyclopedia https://en.wikipedia.org/wiki/Toy_problem

[3] Missionaries and cannibals problem, From Wikipedia, the free encyclopedia https://en.wikipedia.org/wiki/Missionaries_and_cannibals_problem

[4] Cordeschi, Roberto (2006). "Searching in a Maze, in Search of Knowledge: Issues in Early Artificial Intelligence". In Stock, Oliviero; Schaerf, Marco. Reasoning, Action and Interaction in AI Theories and Systems: essays dedicated to Luigia Carlucci Aiello. Lecture Notes in Computer Science. 4155. Berlin/Heidelberg: Springer. pp. 1–23. doi:10.1007/11829263_1. ISBN 978-3-540-37901-0.

[5] Applications of Reinforcement Learning in Real World, 2019 https://towardsdatascience.com/applications-of-reinforcement-learning-in-real-world-1a94955bcd12

[6] I. Arel, C. Liu, T. Urbanik, and A. Kohls, "Reinforcement learning-basedmulti-agent system for network traffic signal control,"IET IntelligentTransport Systems, 2010.

[7] J. Jin, C.Song, H. Li, K. Gai, J.Wang amd W. Zhang. Real-Time Bidding with Multi-Agent Reinforcement Learningin Display Advertising. arXiv preprint arXiv:1802.09756, 2018.

[8] Korf, Richard E (2012). "Research challenges in combinatorial search".

[9] J. Kober, J. A. D. Bagnell, J. Peters. Reinforcement Learning in Robotics: A survey. *Int. J. Robot. Res.* Jul. 2013.

[10] Z. Zhou, X. Li, and R. N. Zare. Optimizing Chemical Reactions with Deep Reinforcement Learning. ACSCentral Science3, 2017.

[11] D. Silver, J.Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M.Lai, A. Bolton, Y. Chen,T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D.Hassabis. Mastering the game of go without human knowledge.Nature, 2017.