REPORT PAPER

Implementation Q learning algorithm to find the path in a maze

Abstract

In this paper, the implementation of reinforcement learning namely Q learning on the agent that looking path in a maze has been discussed. The goal of the experiments is to make the agent learn the shortest way between places in the maze. The less steps it can do, the more reward it accumulates. I did various tests with many hyperparameters.

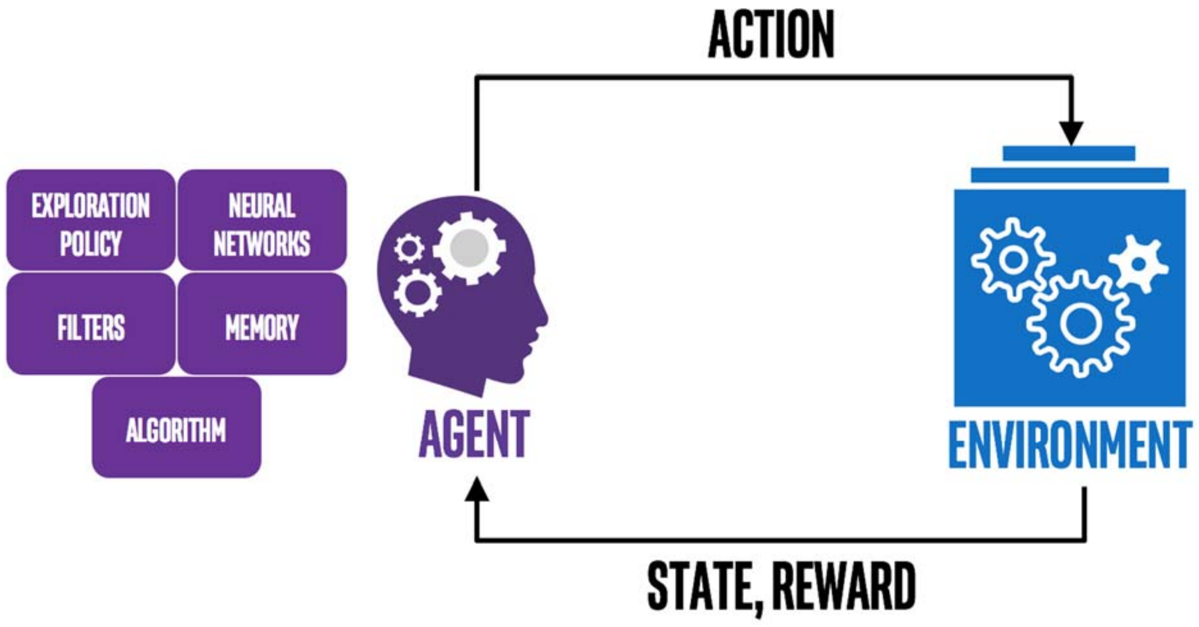
1. Introduction

Reinforcement learning can be understood using the concepts of agents, environments, states, actions and rewards.

- Agent: An **agent** takes actions. The algorithm is the agent.
- Action (A): A is the set of all possible moves the agent can make. An **action** is almost self-explanatory, but it should be noted that agents choose among a list of possible actions. In our case they are: UP, DOWN, LEFT, RIGHT.
- Discount factor: The **discount factor** is multiplied by future rewards as discovered by the agent in order to dampen those rewards' effect on the agent's choice of action. It is designed to make future rewards worth less than immediate rewards; i.e. it enforces a kind of short-term hedonism in the agent.
- Environment: The world through which the agent moves. The environment takes the agent's current state and action as input, and returns as output the agent's reward and its next state. If you are the agent, the environment could be the laws of physics and the rules of society that process your actions and determine the consequences of them.
- State (S): A **state** is a concrete and immediate situation in which the agent finds itself; i.e. a specific place and moment, an instantaneous configuration that puts the agent in relation to other significant things such as tools, obstacles, enemies or prizes. It can the current situation returned by the environment, or any future situation.
- Reward (R): A **reward** is the feedback by which we measure the success or failure of an agent's actions. From any given state, an agent sends output in the form of actions to the environment, and the environment returns the agent's new state (which resulted from acting on the previous state) as well as rewards, if there are any. Rewards can be immediate or delayed. They effectively evaluate the agent's action.
- Policy (π): The **policy** is the strategy that the agent employs to determine the next action based on the current state. It maps states to actions, the actions that promise the highest reward.
- Value (V): The expected long-term return with discount, as opposed to the short-term reward R. $V\pi(s)$ is defined as the expected long-term return of the current state under policy π. We discount rewards, or lower their estimated value, the further into the future they occur. See discount factor.

- Q-value or action-value (Q): **Q-value** is similar to Value, except that it takes an extra parameter, the current action `a`. `Qπ(s, a)` refers to the long-term return of the current state `s`, taking action a under policy `π`. Q maps state-action pairs to rewards.

So environments are functions that transform an action taken in the current state into the next state and a reward; agents are functions that transform the new state and reward into the next action. We can know the agent's function, but we cannot know the function of the environment. It is a black box where we only see the inputs and outputs. Reinforcement learning represents an agent's attempt to approximate the environment's function, such that we can send actions into the black-box environment that maximize the rewards it spits out.



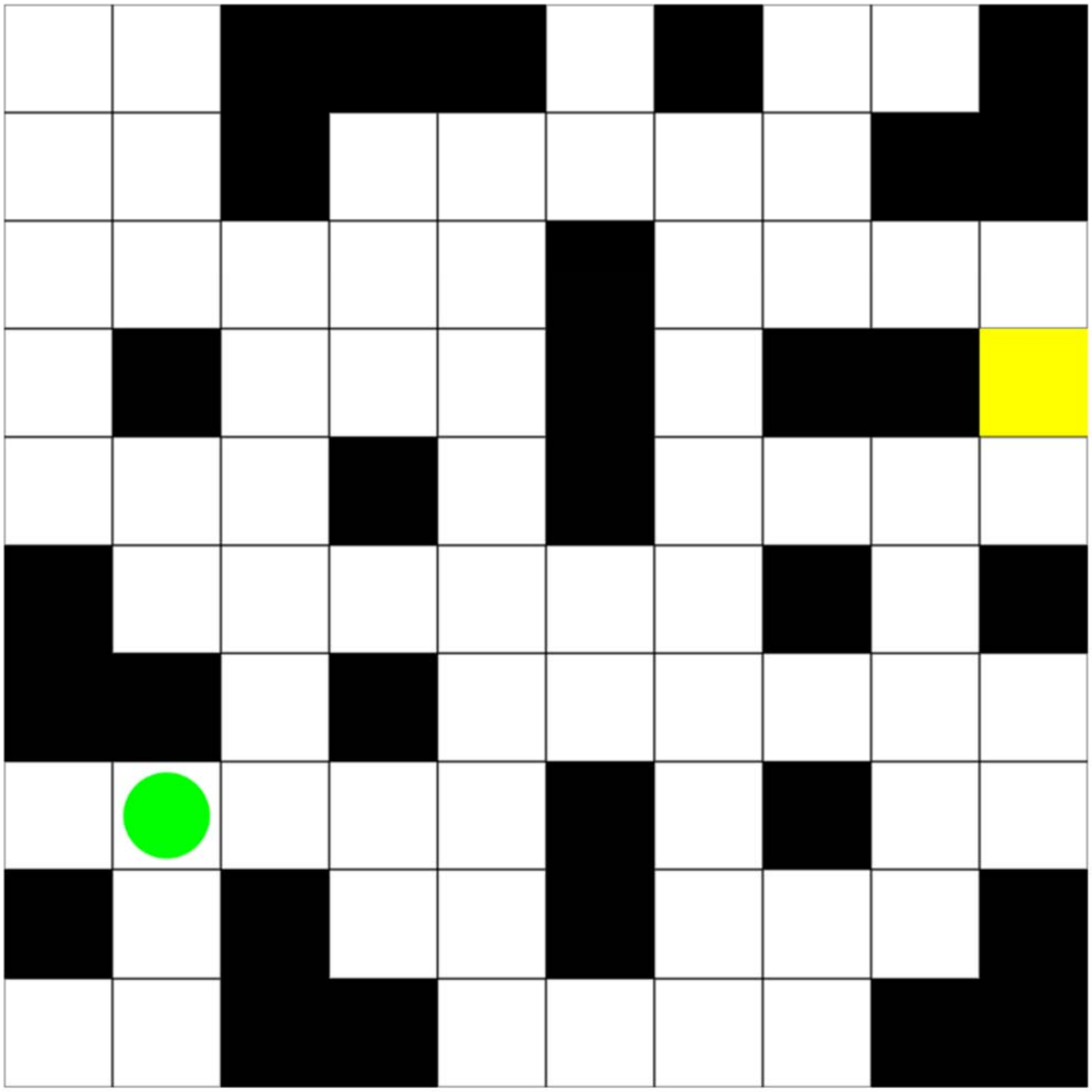
2. Implementation

Q-learning was developed by Christopher John Cornish Hellaby Watkins [1]. According to Watkins, "it provides agents with the capability of learning to act optimally in Markovian domains by experiencing the consequences of actions, without requiring them to build maps of the domains" [2]. In a Markovian domain, *Q* function—the model to be generated using the algorithm—calculates the expected utility for a given finite state *s* and every possible finite action *a*. The agent—which is the robot in this case—selects the optimum action *a* having the highest value of *Q*(*s*, *a*) , this action choosing rule is also called Policy [2]. Initially, the *Q*(*s*, *a*) function values are assumed to be zero. After every training step, the values get updated according to the following equation

$$Q(s,a_t) \leftarrow Q(s,a_t)(1-\alpha)+\alpha(r+\gamma \max Q(s'_t,a)),$$ where

- α – learning rate
- γ – discount rate

For implementation I used browser and JavaScript.



Maze was generated from the array of states, where 0 is an empty space (reward 0) and 1 is a wall (reward is -1).

```
[
          0, 0, 1, 1, 1, 0, 1, 0, 0, 1,
          0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
          0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
          0, 1, 0, 0, 0, 1, 0, 1, 1, 0,
          0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
          1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
          1, 1, 0, 1, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
          1, 0, 1, 0, 0, 1, 0, 0, 0, 1,
          0, 0, 1, 1, 0, 0, 0, 0, 1, 1
        ];
```
Reward at target position is 100.

Learning rate (α) was constant and equal to 0.1

Discount rate (γ) was constant and equal to 0.99

Tradeoff exploration/exploitation rate (ER) has been starts from 1 at first episode and was calculated for every episode according to the following equation

$$ER \leftarrow ER_{min} + (ER_{max} - ER_{min})\, e^{-ER_{decay}Episode},$$

where

- $ER_{min} = 0.01$
- $ER_{max} = 1$
- $ER_{decay} = 0.1$

3. Experimental results

Learning of agent was completed in ~50 episodes with the aforecited parameters. It was best result for this algorithm.

4. Future work

Traps with reward -2 can be added to maze to increase complexity.

5. Conclusion

The implementation of *Q* learning was shown in this paper. It showed the details of the algorithm and parameters.

References

1. Watkins CJ. Learning from delayed rewards. Ph.D. dissertation, Kings's Collenge, London, May 1989.
2. Watkins CJCH, Dayan P. Q-learning, Machine Learning, vol. 8, no. 3, pp. 279–292, May 1992. 10.1007/BF00992698.