Université Jean Monnet

Advanced Machine Learning, MLDM-2

Practical Session 2 Report

Reinforcement Learning

Submitted By:
Karthik Bhaskar

Contents

1 Reinforcement Learning

Reinforcement Learning also called adaptive dynamic programming is a branch in machine learning, In reinforcement learning the system in the environment learns about the environment by performing trail and error actions or methods in that particular environment, for each action performed the reward is given in the form of reinforcement signal which can be 0 or 1 or any real number which may be positive or negative.

The goal is to obtain the maximum number of reward by performing optimal actions which leads to the transition of states in the environment, this process is continued until the environment is reseted or terminal state in the environment is achieved.

Reinforcement learning is used as tool to solve decision making problems in a large and complex Markov Decision Problem(MDP) where the environment is dynamic with infinite timespan, as in MDPs when it is complex and large, due to the curse of dimensionality and curse of modeling[1] makes the classic dynamic programming inefficient therefore the adaptive dynamic programming is used.

The Markov Decision Problem(MDP) is a framework where the system in the environment is linked to Markov chain where the system jumps from one state to other on selecting a particular action in that state in random time and the transition of the state of the system from one state to another state depends on the current state and not the previous visited states.[1]

In MDP, the entities are

- Environment: The space where the agent tries to solve the problem by applying actions
- State: The situation in the environment
- Agent: The explorer of the environment, who tries to solve the problem
- Action: Methods applied in the environment to perform transition of the state
- Policy: The optimal solution chosen to solve the environment or the problem
- Metrics:
  - Discounted Reward: The sum of all rewards achieved over time
  - Average reward: The expected reward expected at each time

1.1 Problem Environment

The problem in the environment we will be solving using reinforcement learning is a NxN grid.

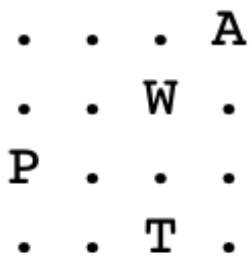

Figure 1: Game Grid

The entities in this game grid are Agent, Empty path, Wardrobe, Treasure and Poison. The goal of this environment is to make the agent find the optimal path to the treasure and rule of the environment is Poison and Treasure are the terminating states and the agent should not traverse through the wall. The rewards are given to the agent, the reward signals are +10 for reaching the goal, -10 reaching the poison state, -8 reaching the wardrobe and moving to the empty state it is -1. The legal actions in the environment are Up, Down, Left and Right.

1.2 Experiment

To solve the above problem and find the optimal policy for the environment, Q-learning algorithm is used.

1.2.1 Q-Learning

Q-Learning is a Reinforcement Learning techniques which is model free technique where it learns about the environment dynamically[2].

This techniques allows the agent to chose optimal policy in the MDP. The agent in the environment executes actions in a particular state and based on the immediate reward it learns the optimal action in that state, the agent tries all the available actions in the particular state and learns the best suit-able action of that state based on the overall average reward.Q-Learning uses a table which includes states and action taken in that state where the row is states and column is the action taken in state. It stores the values in the form of Q-values.

The Q-value is updated based on

$$Q(s,a) = Q(s,a) + \alpha * (r + \gamma * max(Q(s+1,a)) - Q(s,a))$$

Where Q(s,a) is Q value of current state based on the action. r is reward obtained, γ is the discount rate, α is the learning rate, max(Q(s+1,a)) is the maximum expected reward from the next state. The main steps in the algorithm are,

- Q values are initialized to zero at start
- Random action are chosen at start to explore the environment to greedily increase Q-value
- Reward achieved from the action is updated to the Q-table
- In the exploitation stage the action with max Q-value in that stage is chosen to increase the reward and the Q-table is updated with new Q-value.

1.2.2 Results

This algorithm is implemented on 4x4 and 5x5 grid. The grid was created to implement the environment using text based matrix. The algorithm was executed with 30 Episodes with 100 time steps and the value of the epsilon was chosen to be 0.1 for exploration, $\gamma = 0.8$, the total reward of each episode was calculated. The agent started to find the optimal path after 20 episodes in order to obtain positive reward and to choose optimal path to the target.

```
Episode 1 reward: -13
Episode 2 reward: -14
Episode 3 reward: 2
Episode 4 reward: -11
Episode 5 reward: -30
Episode 6 reward: -16
Episode 7 reward: 5
Episode 8 reward: -1
Episode 9 reward: 1
Episode 10 reward: 1
Episode 11 reward: 7
Episode 12 reward: 7
Episode 13 reward: 7
Episode 14 reward: 7
Episode 15 reward: 7
Episode 16 reward: 7
Episode 17 reward: 7
Episode 18 reward: 7
Episode 19 reward: 7
Episode 20 reward: -9
Episode 21 reward: 7
Episode 22 reward: 7
Episode 23 reward: 7
Episode 24 reward: 7
Episode 25 reward: 7
Episode 26 reward: 7
Episode 27 reward: 7
Episode 28 reward: 7
Episode 29 reward: 7
Episode 30 reward: 6
```

Figure 2: Result

In the implementation code, the movement of the agent can be seen. As the environment grid is small, good results were obtained, as the environment grid grows it is hard to obtain the optimal policy and calculating the Q-values in this manner becomes in-efficient, this can solved using neural network. This implementation was inspired from this blog [3]

2 Conclusion

To conclude Reinforcement Learning is a machine learning technique used to solve a problem by trail and error method to find the optimal solution.

References

[1] Abhijit Gosavi. Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing*, 21(2):178–192, 2009.

[2] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[3] Solving an mdp with q-learning from scratch [U+200A] — [U+200A] deep reinforcement learning for hackers (part 1). https://bit.ly/2HDIlwU.