Markov Decision Process

Felicia Fryer
Georgia Institute of Technology
ffryer3@gatech.edu
April 14, 2019

Abstract

This paper examines two Markov Decision Processes (MDP), Stock Trading and Frozen Lake. Each MDP is solved using both value iteration and policy iteration and evaluates the performances of each method. Also, the MDP problem will be solve using Q-Learning which is a reinforcement learning algorithm.

I. INTRODUCTION

Markov Decision Process (MDP) is derived from a concept known as *Markov Chains*. Markov Chains is a stochastic process that has a finite number of states and randomly transitions to the next step. A mathematician named Richard Bellman used the concept of Markov Chains to create Markov Decision Process. The Markov Decision Process problems that are examined in this paper are Frozen Lake and Stock Trading. I choose the Frozen Lake environment because it is a simple way to showcase how an agent moves within a grid world with obstacles in a stochastic manner. The Frozen Lake scenario is a problem about controlling an agent. The stock trading problem is a project that I am familiar with and I have performed in the past in another GaTech's course, Machine Learning for Trading. The stock trading problem is a prediction problem, in which the MDP algorithm will predict future stock values based on historical prices. The stock market is not predictable but MDP can help to mitigate losses

II. METHOLOGY

A. Markov Decision Process

MDP can be describe as when an agent chooses an action. Each step has a transition probability for the action taken. State transitions has an associated reward, and the goal of the agent is to maximize the reward. We will explore two methods to maximize the reward and find the optimal policy for an MDP process. The first method is known a value-iteration. Value-iteration determines the optimal state value function, V(s), by iteratively improving the estimated V(s). The pseudo-code for the algorithm for value-iteration convergence is shown below:

Initialize $V(s)$ to arbitrary values
Repeat
  For all $s \in S$
    For all $a \in \mathcal{A}$
      $Q(s,a) \leftarrow E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a)V(s')$
    $V(s) \leftarrow \max_a Q(s,a)$
Until $V(s)$ converge

The second method is policy-iteration. Policy-iterations re-define the policy at each step and compute the value according to this new policy until the policy converges [1]. The pseudo-code for the algorithm for policy-iteration convergence is shown below:

Initialize a policy π' arbitrarily
Repeat
  $\pi \leftarrow \pi'$
  Compute the values using π by
    solving the linear equations
      $V^{\pi}(s) = E[r|s,\pi(s)] + \gamma \sum_{s' \in S} P(s'|s,\pi(s))V^{\pi}(s')$
  Improve the policy at each state
    $\pi'(s) \leftarrow \arg\max_a (E[r|s,a] + \gamma \sum_{s' \in S} P(s'|s,a)V^{\pi}(s'))$
Until $\pi = \pi'$

B. Q-Learning

Q-Learning is a model-free reinforcement learning algorithm [2]. When the agent does not know the transition probabilities and the rewards, Q-learning is a good model to use. The agent will experience each state and each transition at least once to know the rewards. Q-learning finds a policy that maximizes the total reward. The pseudo-code for the Q-Learning algorithm is shown below:

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q
    Take action a, observe r, s'
    Update
      $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
    $s \leftarrow s'$;
  Until s is terminal

C. Markov Decision Processes Problem

1) Frozen Lake

The Frozen Lake is a stochastic environment taken from OpenAI Gym. This environment is a grid world where an agent can move around the grid tiles. Some tiles are considered slippery; thus, the agent may not go in the intended direction at times. Some of the tiles are passable and others are not. The agent is rewarded for finding a walkable path to a good tile [3]. It has 16 states and 4 actions. The Frozen Lake environment is an interesting problem to solve because it mimics the action of a robot in a maze, which are fun problems to solve.

2) Stock Trading

The stock trading problem is from the Machine Learning for Trading course I took last semester. I use reinforcement learning to find the optimal policy for trading stocks. For this project I will use stock symbol 'GSPC' which is the ticker symbol for the price index of the S&P 500 for between dates 9-18-1950 and 9-18-2018. There are three states which is determine by the Bollinger band stock indicator. The three states are as follows: state (1) if return is higher than average, state (2) if return is lower than average, and state (3) if neither is the case. There are three actions, buy, hold, short. The reward is the sharpe ratio which measures the performance of a portfolio. The stock MDP problem converges when the reward is at the maximum value which in this case is the sharpe ratio.

D. Python Libraries

The Python libraries that was used for this project are as follows: Numpy, Gym, Pandas, Matplotlib, Random, Collections, Wrappers.

III. RESULTS AND DISCUSSION

A. Frozen Lake

1) How many iterations does it take to converge for both value and policy-iteration methods?

I performed 10 experiments for both value-iteration method and policy-iteration method. The results are below.

Experiments	Policy-Iteration (Score)	Policy-Iteration (Convergence Step)	Value-Iteration (Score)	Value-Iteration (Iteration Count)
1	0.72	3	0.74	1373
2	0.76	3	0.72	1373
3	0.73	7	0.75	1373
4	0.67	5	0.73	1373
5	0.7	4	0.76	1373
6	0.69	7	0.76	1373
7	0.65	7	0.75	1373
8	0.73	4	0.72	1373
9	0.73	7	0.74	1373
10	0.72	7	0.73	1373

2) *Which method converges faster and why?*
On average the policy-iteration had a score of 0.78, and the value-iteration score had a score of 0.74. The policy-iteration scored higher. In general, the policy-iteration method converge faster than the value-iteration method due to the fact that the policy-iteration method because it guarantees to converge to the optimal policy and takes less iterations to do so [1].

3) *Do they converge to the same answer?*
Yes both policy-iteration and value-iteration converge to the same answer.

The optimal policy for the policy-iteration method is
`array([0., 3., 3., 3., 0., 0., 0., 0., 3., 1., 0., 0., 0., 2., 1., 0.])`

The optimal policy for the value-iteration method is
`array([0., 3., 3., 3., 0., 0., 0., 0., 3., 1., 0., 0., 0., 2., 1., 0.])`

4) *How did the number of states affect things, if at all?*
For the Frozen Lake environment, the number of states is fixed at 16 states.

5) *How does the Q-Learning perform compared to the other methods?*
I performed 10 episodes of Q-Learning experiments. The table below is a graph of the steps of convergence for Q-Learning algorithm. Since the Q-Learning algorithm is a model-free method, it takes slightly longer to find the optimal policy. With the Q-Learning algorithm, the agent is learning the grid world be "trial and error" and exploring the grid world in a random search.

Episodes	Steps to Convergence
1	11
2	45
3	17
4	52
5	25
6	12
7	34
8	24
9	70
10	37

6) *What exploration strategies did you choose and did some work better than others?*
The exploration parameters that were used are as follows:

- exploration rate (epsilon) = 1.0
- exploration probability at start (maximum epsilon) = 1.0
- minimum exploration probability (minimum epsilon) = 0.01
- exponential decay rate for exploration probability (decay rate) = 0.005

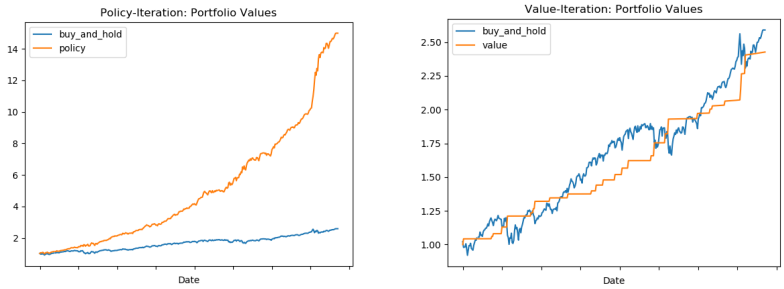
B. Stock Trading

1) *How many iterations does it take to converge for both value and policy-iteration methods?*
The table below shows the iteration count for both the policy-iteration method and the value-iteration method.

Experiments	Policy-Iteration (Sharpe Ratio)	Policy-Iteration (Iteration Count)	Value-Iteration (Sharpe Ratio)	Value-Iteration (Iteration Count)
1	.176	4	0.153	13
2	.164	7	0.105	34
3	.182	8	0.152	6
4	.154	7	0.164	11
5	.193	6	0.122	4

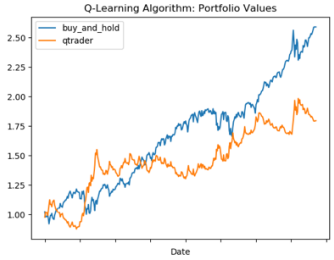
2) *Which method converges faster and why?*
The policy-iteration converge faster. The policy-iteration method converges faster because it uses a greedy approach.

3) *Do they converge to the same answer?*
The different methods do not converge to the same answer. Below are the graphs for the policy-iteration and value-iteration portfolio values. From the graph the policy-iteration portfolio performs better than the benchmark buy-and-hold portfolio, while the value-iteration portfolio performs just as good as the buy-and-hold portfolio.



4) *How did the number of states affect things, if at all?*
The number of states did not affect things.

5) *How does the Q-Learning perform compared to the other methods?*
The Q-Learning does not perform as well as the policy-iteration method.



6) *What exploration strategies did you choose and did some work better than others?*
For the Q-Learning algorithm, I implemented a dynamic programming method which "hallucinates" the transition and reward model.
The exploration parameters that were used are as follows:

- exploration rate (epsilon) = 1.0
- exponential decay rate for exploration probability (decay rate) = 0.8

IV. CONCLUSION

In conclusion, the Markov Decision Process is a good tool to use for both deterministic and stochastic problem solving. For this project I have used the MDP in an agent-like environment that is commonly seen in gaming and have used it for stock trading. Markov Decision Process is versatile and robust enough to for a variety of problem-solving scenarios. According to class lectures the policy-iteration method "convergence is exact and complete in a finite time", and the policy-iteration method converges at least as fast as value-iteration method.

V. REFERENCES

[1] https://medium.com/@m.alzantot/deep-reinforcement-learning-demysitifed-episode-2-policy-iteration-value-iteration-and-q-978f9e89ddaa

[2] https://en.m.wikipedia.org/wiki/Q-learning

[3] https://gym.openai.com/envs/FrozenLake8x8-v0/