Markov Decision Processes
CS 7641

Abel Aguilar
aaguilar61@gatech.edu

1 INTRODUCTION TO PROBLEMS

In this assignment I choose two different Markov Decision Process problems to see how good they are at developing a policy for game specific games. The two problems I chose were the Frozen Lake and Black Jack. Frozen Lake is a game where a player starts in the position 0,0 of a grid and in the case of the game I chose the reward is at position 3,3 of a 4x4 grid. The actions our player can take are Up, Down, Left, Right to move between spaces. The goal of the game is for the player to reach the goal without falling into any holes along the way. However, the floor is slippery and there is a chance that that player moves in a perpendicular movement from what they intended in any given move. This game is interesting for a number of different reasons, one reason is that game and state space is very small. It only has a 4x4 space which translates to 16 states. This takes a lot of complexity out of solving this problem. Another interesting aspect of this game is the reward structure. In this environment every square except the reward square gives 0 as a reward including the holes. This structure makes it so that our character is in no rush to finish the game and it can sometimes get stuck in a loop because it is not being punished for not finishing the game. Also because the holes are also 0, while it will still try to avoid the hole because that means ending the game with no points, it doesn't have as strong of an avoidance of the hole as it should.

The second game I chose was Black Jack. In this game the dealer starts with a card face up and the player gets two cards. The player then has two actions, hit or stay. The goal of the game is to get as close to 21 as possible without going over. Each card has one value except for an Ace which can be used as either a One or an Eleven. After our player stays, the deal then continues to pick cards until they go over 17. The winner is the person who gets closest to 21 without going over. What makes this game interesting is it has a large number of states. Every possible combination of cards our player can get plus the dealer's face up card determines the states. They sum up to around 300 different possible states. This adds a bit of complexity quickly determining when to take risks and when

not to. Another interesting aspect of this game is the reward system. In this game our agent gets a +1 for winning a game, a +1.5 for winning in with a natural (a 10 and an Ace), a -1 for a loss, and a 0 for a draw. What is interesting about this reward system is a natural win with the highest reward even though this happens before our agent can make any moves so this reward does not really serve to change how our agent behaves. It is also interesting that no reward, either positive or negative, is given for a draw. This will make our agent more conservative and take less risk to win a game and it might try to settle for a draw.

2 EXPERIMENTAL METHODOLOGY

In order to test each game, I will be creating policies to try to solve each game using value iteration, policy iteration, and q learning. Some of the things I will be looking for in each test is how quickly each policy coverages, and how optimal the policies are once coveraged.

I will also be using visualization to actually see how each policy plays out on the actual game and compare the three different policy planning methods. For this heats maps and policy grids will have a better understanding of what the policy believes we should do.

3 VALUE ITERATION AND POLICY ITERATION

3.1 Frozen Lake Value Iteration

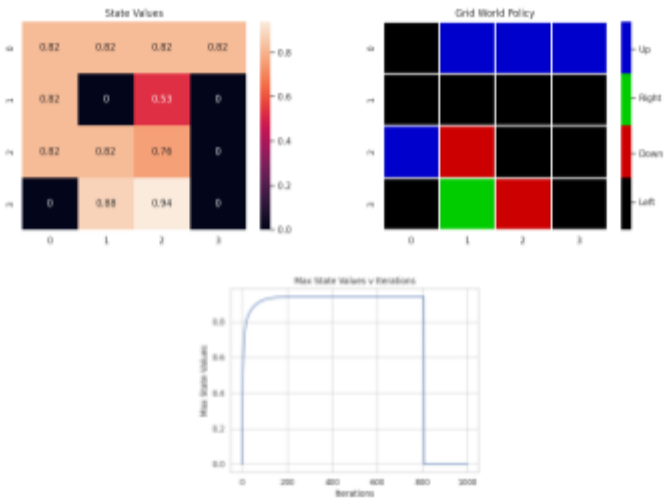

Figure 1—Value Iteration Graphs FL

With value iteration you can see that since there is not much risk in staying in the game longer, many moves we would have thought to be obvious our agent is taking the more safe options. For example, in space (0,1), we could expect our agent to travel down, but instead it travels left to avoid the possibility of a slip and accidentally falling into the hole. This is because it is relying on a slip to move it down since it doesn't lose anything when running into walls or takes a long time in the game. While this policy is not optimal to getting to the goal each in a short period of time this policy does get to the goal a majority of the time. The benefits here is also that it converges very quickly, 0.08s run time. As the space increases the policy looks for similar solutions but this creates a very long game since our policy relies on slips going our way.

3.2 Frozen Lake Policy Iteration

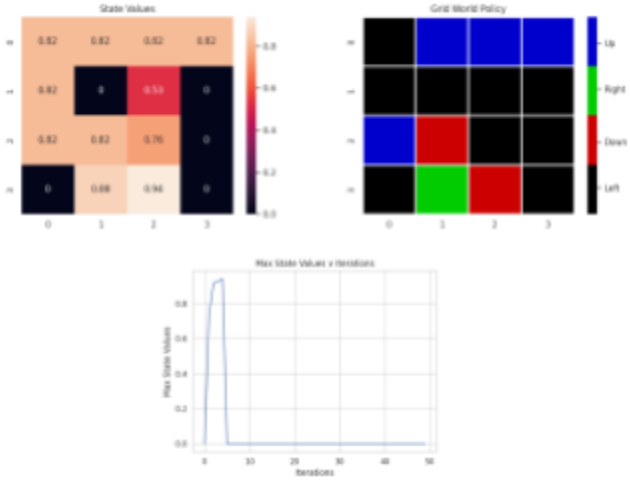

Figure 2—Policy Iteration Graphs FL

With policy iteration we can see that because of the way the award structure is set up policy iteration reaches the same optimal policy but in fewer iterations. However, this is a more computationally expensive method. The run time here is 0.13s, which does not seem like a lot but as the problem scales policy iteration takes longer and longer. Again we see that because of the reward structure our policy does not take many risks and makes a larger scale problem much more time consuming to solve simply because on solution relies on slips going our way. This is also clear when looking at the heat map.

3.3 BlackJack Value Iteration

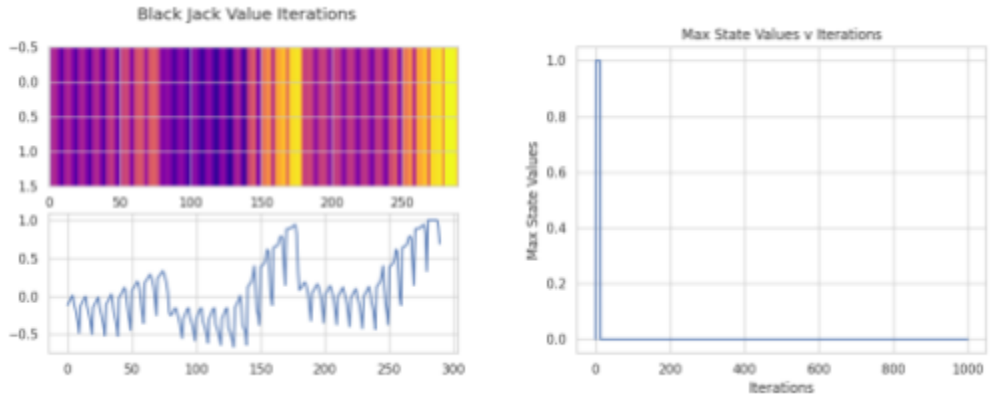

Figure 3—Value Iteration Graphs BJ

When looking at how the blackjack policy when value iteration converges we see that the lower the sum of cards you have the higher likelihood that our agent calls for a hit, which makes sense. One interesting thing that we learn is that as the dealer's card gets higher the more risks we are willing to take to win. We also see that convergence is reached very quickly both in terms of wall time and iterations. Here convergence is reached in 0.02s. It looks with this type of game as the state space increases the policies, since this is mostly a game of chance the type of risk that our policy would take stays the same.

3.4 BlackJack Policy Iteration

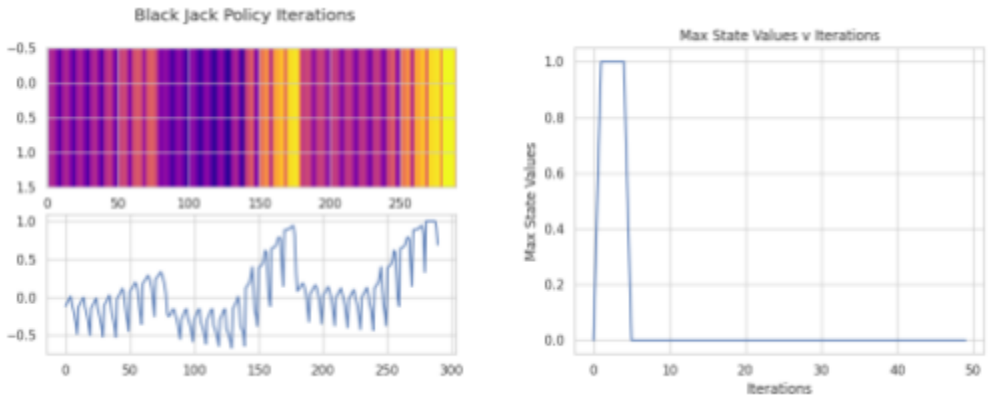

Figure 4—Policy Iteration Graphs BJ

Again here we see that the policy reached by Policy Iteration is very similar to what was reached by value iteration. The main difference we see here again is that policy iteration is much more computationally heavy then value iterations.

The wall time here is 0.30s. This also means that as the state space grows the wall time is only going to get more expensive. However, I don't believe that policy would change very much in a larger state space other than maybe shifting up or down depending on what the winning value in blackjack changes too.

4 Q LEARNING

4.1 Frozen Lake Q Learner

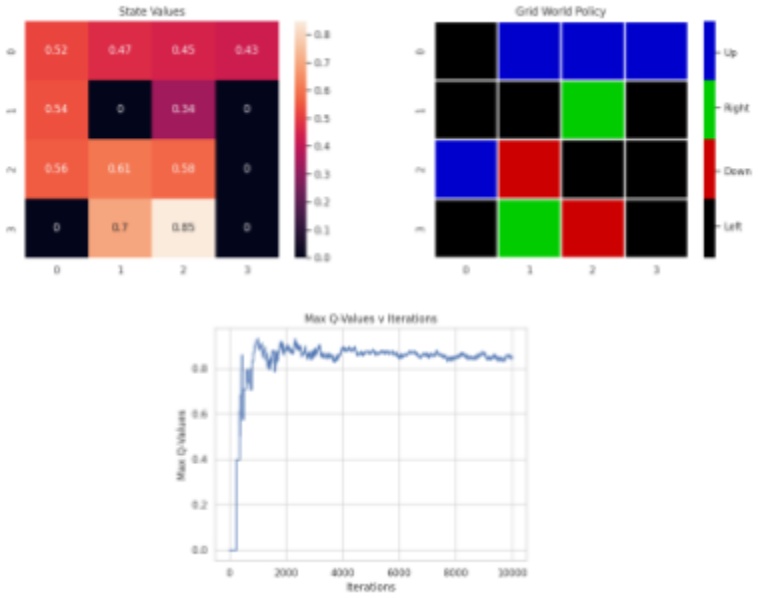

Figure 5—Q Learner Graphs FL

The immediate difference between our q learner and our other policies is our q learner even though it ends up at a very similar policy, it is a lot less sure about each of its decisions as we can see through our heat map. Also our q learner never really converges, as we can see in the Max Q-Values vs Iterations. It looks like the policy doesn't really change much but it is not as sure as our other methods. Another difference is the wall time, q learning takes much longer at 3.66s of wall time. As the state space grows again because of the way the reward structure is set up, not much should change in terms of policy, it will always play it safe and rely on slips as much as possible. However, the wall time will increase much faster for q learning then any other method as the state space increases.

4.2 BlackJack Q Learner

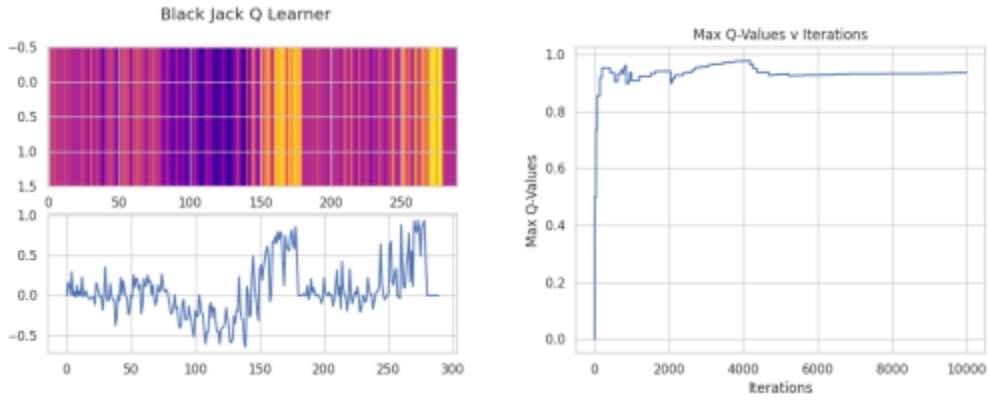

Figure 6—Q Learner Graphs BJ

One interesting difference we see between q learning and our other methods is that this method takes much more risks than our other policies. We also see that again convergence is never really reached but the changes stabilize in our final iterations which is effectively a convergence. Again we see that wall time is much higher for q learning coming in at 1.12s. It also looks like with this game in particular because of the unsure nature of q learning as the state space grows, it is likely to take more and more risks and increase very quickly in terms of wall times.

5 REFERENCES

1. "Bettermdptools." GitHub, github.com/jlm429/bettermdptools/blob/master/readme.md. Accessed 26 Nov. 2023.
2. "BlackJack." GitHub, github.com/jlm429/bettermdptools/blob/master/examples/blackjack.py. Accessed 26 Nov. 2023.
3. Farama-Foundation. "Farama-Foundation/Gymnasium: An API Standard for Single-Agent Reinforcement Learning Environments, with Popular Reference Environments and Related Utilities (Formerly Gym)." GitHub, github.com/Farama-Foundation/Gymnasium. Accessed 26 Nov. 2023.
4. "Frozen Lake." GitHub, github.com/jlm429/bettermdptools/blob/master/examples/frozen_lake.py. Accessed 26 Nov. 2023.
5. "Gymnasium Documentation." Blackjack - Gymnasium Documentation, gymnasium.farama.org/environments/toy_text/blackjack/. Accessed 26 Nov. 2023.
6. "Gymnasium Documentation." Frozen Lake - Gymnasium Documentation, gymnasium.farama.org/environments/toy_text/frozen_lake/. Accessed 26 Nov. 2023.
7. "Plots." GitHub, github.com/jlm429/bettermdptools/blob/master/examples/plots.py. Accessed 26 Nov. 2023.
8. Zhang, Jeremy. "Reinforcement Learning-Solving Blackjack." Medium, Towards Data Science, 16 June 2019, towardsdatascience.com/reinforcement-learning-solving-blackjack-5e31a7fb371f.