Markov Decision Processes

Andrew Shi

CS 7641

November 26, 2023

1. Introduction

Markov Decision Processes (MDPs) are a framework for mathematically-backed models for decision learning. In an MDP, an environment is modeled with discrete states and a set of transition actions which may or may not be stochastic. In this analysis, we will examine the behavior of two different MDPs, one of a grid-world nature, and one of a non-grid-world nature. The following reinforcement learning techniques will be used: value iteration, policy iteration, and q-learning.

2. Problems

2.1. Frozen Lake

Frozen lake is a grid world MDP taking place on a square grid with a start square, a goal square, and a number of hole squares. The problem is to travel from the start to goal square while avoiding the holes using an action space comprising of left, up, right, and down movements. The problem given was performed on a 24 x 24 grid, resulting in a total of 576 observation states. While deterministic in nature, the problem has been modified to include stochasticity; namely, any action in a direction has a one-third chance to move in either of the adjacent directions. Frozen lake represents a relatively simple and easy-to-visualize MDP, as grid worlds usually are, where actions represent adjacent transitions in the transition matrix. The rewards are defaulted to the following, but will be altered in future sections: Goal has reward +1, Hole has reward 0, and Frozen has reward 0.

2.2. Blackjack

Blackjack an MDP following the namesake card game. The observation space on this problem is a bit more complicated, though, as the player's cards can be split into hard – meaning a usable ace is included – and soft – meaning no usable ace is included. The hard hands vary from 4 to 21, denoted by H4 – H21, and the soft hands vary from 12 to 21, denoted by S12 – S21, totaling 28 states. Including a blackjack hand, this there are a total of 29 player hands. For each of these, there are 10 values the dealer's card may take; therefore, there are 290 states in all.

As with frozen lake, the blackjack MDP is a stochastic one. When a "Hit" action is submitted by the agent, 10 different stages are possible, as the card is drawn randomly from a standard deck with replacement. Because of that, it serves the agent no purpose to keep track of previous states, and each state is independent of those before it. The reward structure is defaulted as follows: Win has reward +1, Lose has reward -1, and Draw has reward 0.

3. Frozen Lake

Using both value iteration, policy iteration, and q learning, policies can be obtained after a number of iterations. The default parameters of each algorithm are shown below:

Parameter	Value	Parameter	Values
Gamma (VI)	1.0	Gamma (QL)	0.99
n_iters (VI)	1000	Alpha Sched (QL)	0.5 : 0.5 : 0.01
Theta (VI)	1E-10	Epsilon Sched (QL)	1.0 : 0.9 : 0.1
Gamma (PI)	1.0	n_episodes (QL)	10000
n_iters (PI)	50		
Theta (PI)	1E-10		

3.1. Value Iteration

The total change in utility, state values, and policy are shown in figure 2 following value iteration. Total change in utility was chosen as a measure of convergence, as minimal total adjustment in utility represents steady state utility values, and therefore convergence.

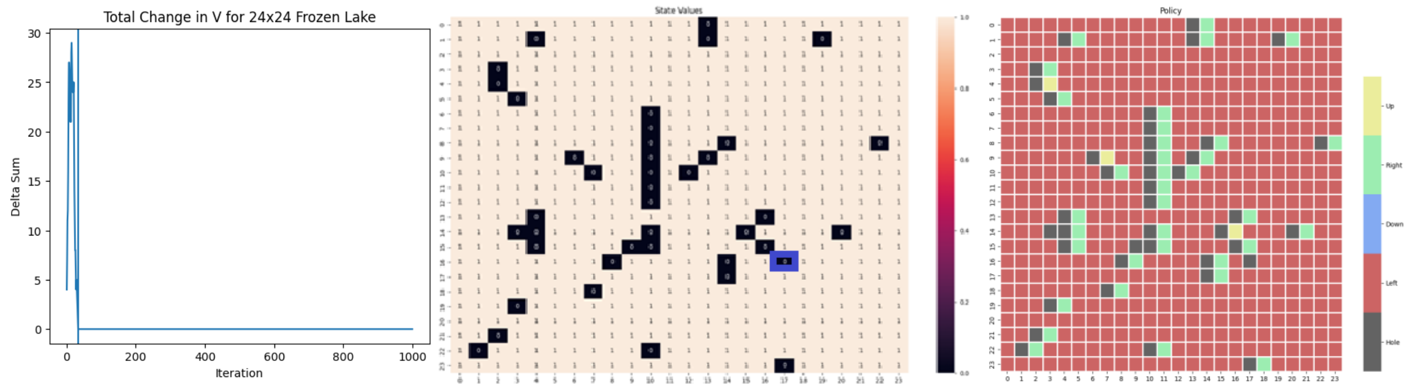

Figure 2. Total change in utility, Value of each state, Policy suggestion at each state for deterministic frozen lake MDP with default rewards. Goal state outlined in blue.

This series of plots features a number of oddities. First, note that every state that is not a hole or goal has value 1. This makes sense, as the path to the goal, whose value is 1, propagates as a result of value iteration, and eventually reaches every state in the grid. The absence of negative rewards for excess steps – a result of the reward structure – and the possibility of "accidentally" stepping into a hole – a result of lack of stochasticity – mean that each state in theory should have the same utility as the goal state. Second, notice that the policy suggested by value iteration is not exactly towards the goal state; in fact, the policy suggests that in most states, moving left is best, unless that results in moving directly into a hole. This behavior is of course also attributed to the flawed reward structure of the default problem. Because any given step onto a frozen square does not result in a negative reward, there is no reason for the agent to even attempt moving towards an arbitrary goal state. Instead, the safest action – left actions – are spammed. To fix this, we alter the reward structure to make the goal +10, the holes -1, and all other steps -0.04. We also introduce stochasticity by making the ice slippery. Figure 3 shows the total change in utility, the state values, and policy.

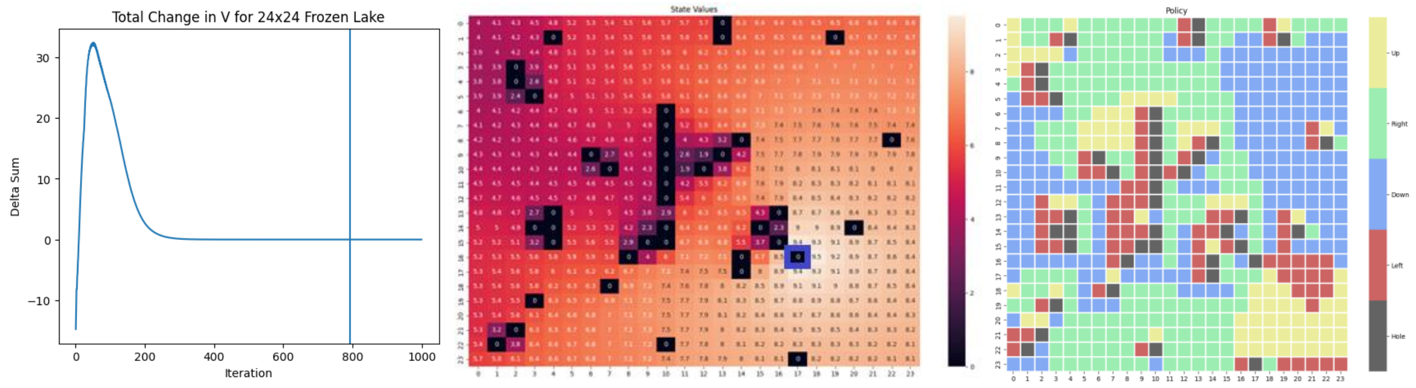

Figure 3. Total change in utility, Value of each state, Policy suggestion at each state after value iteration for stochastic frozen lake MDP with adjusted rewards. Goal state outlined in blue.

Already, we see results that more accurately line with what we would expect. The vertical line in the leftmost plot shows the iteration where convergence is reached, which, in this case is iteration number 792. Convergence is defined by value iteration as when the maximum change in utility between iterations is less that the theta convergence threshold, which is defaulted to 1 * 10^{-10}. The changes are shown to be slightly negative in the first few dozen iterations, likely due a number of holes in close proximity to the goal state. Slowly, the utility decreases as distance to the goal state increases, and the decrease is accelerated by nearby holes that the agent could "accidentally" step into. The policy shows an interesting avoidance to the holes when adjacent to the current state. In these states, the precedence is to always make the move in the opposite direction of the hole. This is due to the ratio of negative reward of holes to the positive reward of the goal. The inclusion of these rewards along with the step reward practically eliminate the problem of "useless moves" for the default reward MDP. Convergence was reached at iteration 792, though the majority of value changes were made within the first 200 iterations.

3.2. Policy Iteration

The total change in utility, state values, and policy are shown in figure 4 following policy iteration.

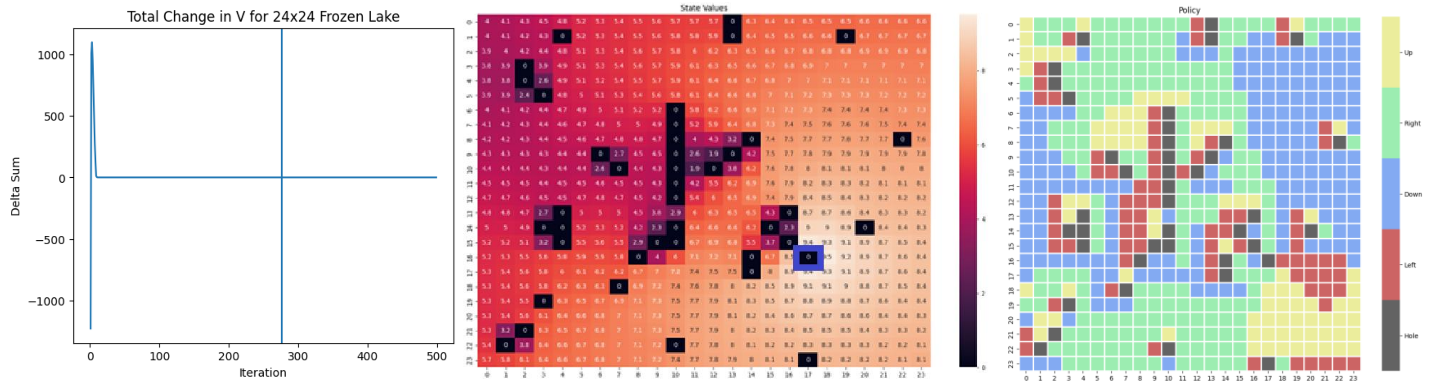

Figure 4. Total change in utility, Value of each state, Policy suggestion at each state after policy iteration for stochastic frozen lake MDP with default rewards. Goal state outlined in blue.

Note that while the number of iterations was increased from 50 to 500 in order to reach the convergence threshold of $1 * 10^{-10}$. In fact, the number of iterations where large changes are made is very low, only about 3. 273 more iterations are included to refine the utility until the threshold is met. Compared to the value iteration, policy iteration operates in far fewer iterations, mainly due to policy iteration's ability to make large jumps in the space of state values based on current policy evaluations. Of course, because the initial policy is random, there will be slight variations in the number of iterations to reach convergence. Furthermore, only a total of 4 inconsistencies exists between the two policy plots. However, each iteration takes significantly longer time – the 276 total iterations took 294.19 seconds, compared to the 5.37 seconds for the 792 iterations of value iteration, to produce near identical policies.

3.3. Q learning

The total change in utility, state values, and policy are shown in figure 5 following q learning.

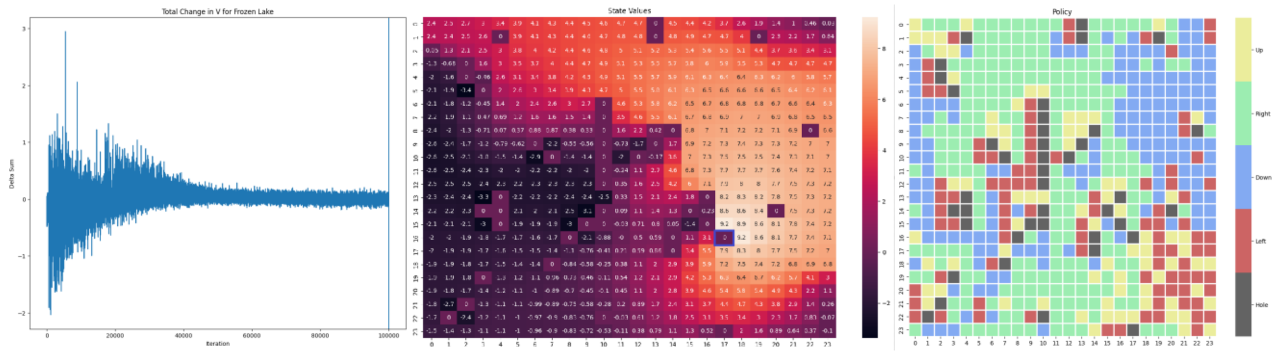

Figure 5. Total change in utility, Value of each state, Policy suggestion at each state after Q learning for stochastic frozen lake MDP with default rewards. Goal state outlined in blue.

Q learning is a model free reinforcement learning algorithm that keeps track of the utility of state-action pairs based on transition pairs indicating the reward and next state of a given state and action pair. Q learning was run on frozen lake for 100,000 episodes with 100,000 maximum steps per episode. This value was set as an absolute high limit, to prevent reward truncation to prevent inaccuracies. Gamma was also set to 1, as q learning seems to dissipate values more quickly than others due to lacking model. We also notice that convergence in the sense that changes in state value are minimal does not hold for q learning as it does for other algorithms. Alpha, which indicates learning rate, begins at 0.5, and decreases towards 0.01 over a number of episodes. With a minimal alpha value, which is set to 0.01 in this instance, each q value will continue to update indefinitely, which is indicated by the prolonged tail of change not being below a certain value.

It is vital to note that the reward structure was changed for holes to be -5. When run with hole reward of -1, the goal reward fails to propagate to the start state. Considering the goal is 33 tiles away from the start state, which totals a -1.32 reward to arrive there, it seems the agent finds greater rewards by jumping directly into a hole, a total reward of -1. To remedy this, the agent was dissuaded from entering holes, and it eventually does progress towards the goal state. The suggested policy was quite different from that of the first two algorithms, with more patches of uncertainty. For example, the bottom right corner, though rare to reach, seems to be a mix of up and left, especially compared to the first two algorithms. Since q learning only marks optimal actions given certain states, and both up and left move optimally close to the goal state, there seems to be a somewhat random selection between the two.

3.4. Hyperparameter adjustments

A number of hyperparameters are considered for the execution of value iteration and policy iteration, namely, gamma and theta, both of which have been discussed to some extent.

Gamma, or the discount factor, is the rate at which values are propagated across the observation space. A high gamma allows for a reward to be broadcast infinitely far without decay, while a low gamma allows for a reward to only be seen locally. Figure 6 shows the state values and policy suggestions following value iteration as gamma decreases.

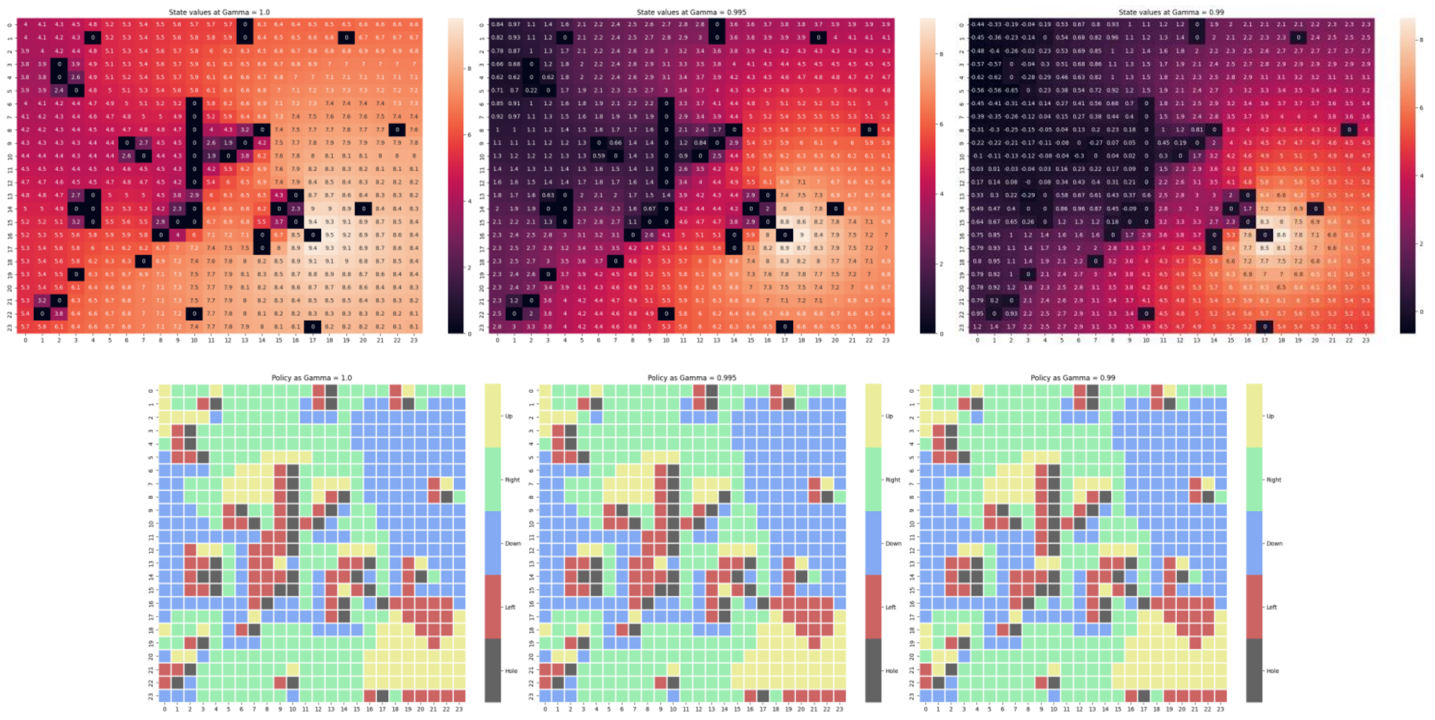

Figure 6. Value of each state and Policy suggestions for gamma = 1.0, 0.995, and 0.99.

Stark changes are observed in the state value plots as gamma is reduced. As gamma is reduced, even as slightly as by 0.05, it is clear that the positive – and negative – values do not propogate quite as far. The discount factors effect on the search can be described as exponential, as each adjacent square away from the goal state experiences a discount, first 0.995, then 0.995^2, then 0.995^3, etc. Coupled with the -0.04 reward of taking a step onto a frozen state, we begin to see negative utilities to a number of states near the starting square for gamma = 0.99.

The policy suggestion plots look quite similar, barring minor changes. It is expected that gamma does not have significant effects in the policies produced, as all transitions are effected equally. The number of iterations to reach convergence is shown in figure 7. It is evident that the decrease in discount factor means convergence is reached in fewer iterations, likely due to the constrained propagation of rewards.

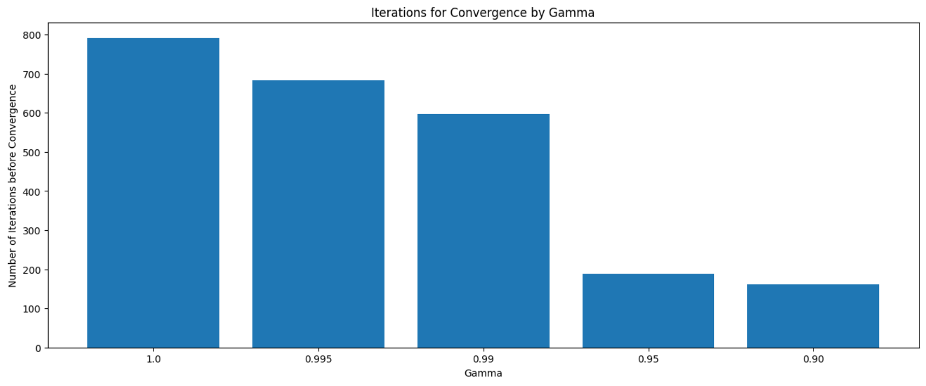

Figure 7. Number of iterations to reach convergence by gamma

Theta, as mentioned before, is the convergence margin. When the maximum change of a state over a single iteration is lower than theta, convergence is considered reached. Figure 8 shows the state values and policy suggestions following value iteration as theta varies.

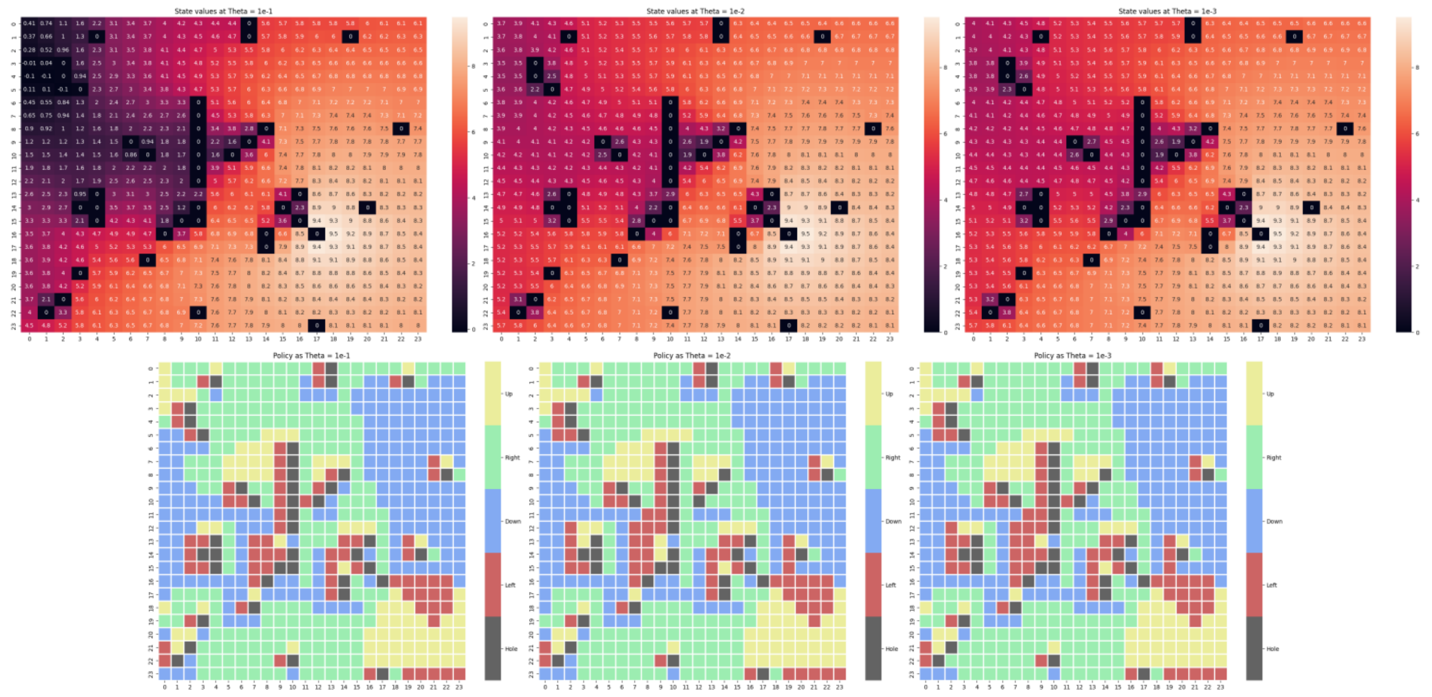

Figure 8. Value of each state and Policy suggestions for theta = 1e-1, 1e-2, 1e-3

When theta is only 1e-1, the top left corner of the plot has far lower state values than those of the others. The higher theta allows for convergence to be considered before the reward is propagated to that area, which is nonoptimal. We see that quickly, as theta decreases to 1e-2, the state value plot seems to converge closely on the state value plot found with theta = 1e-10 from before. The total absolute difference between theta = 1e-1 and theta = 1e-10 is 507.881. However, for theta = 1e-2, the total absolute difference drops to 28.783. The policy suggestion plots show that though a near optimal solution is found using each of the thetas, though some dissimilarities appear with the higher thetas. The higher theta values simply represent earlier iterations of the plots to be produced by further iterations. The number of iterations to reach convergence as theta increases is shown in figure 9. It makes sense that the lower theta values take more iterations to reach, given the previous remarks.

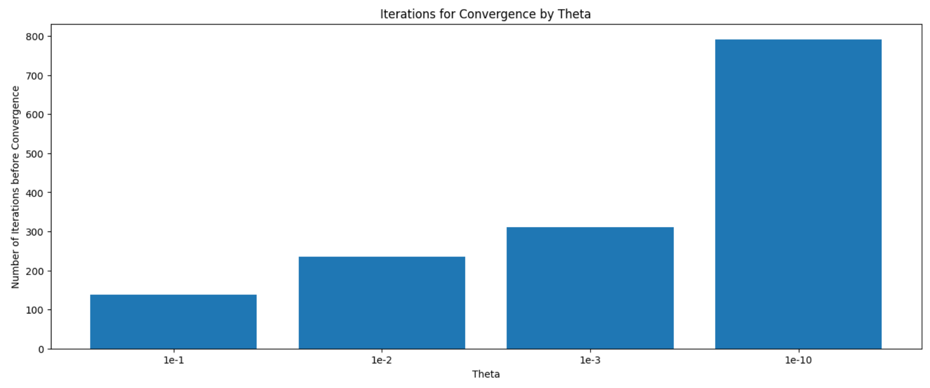

Figure 9. Number of iterations to reach convergence by theta

4. Blackjack

4.1. Value Iteration

The total change in utility, state values, and policy are shown in figure 10 following value iteration.

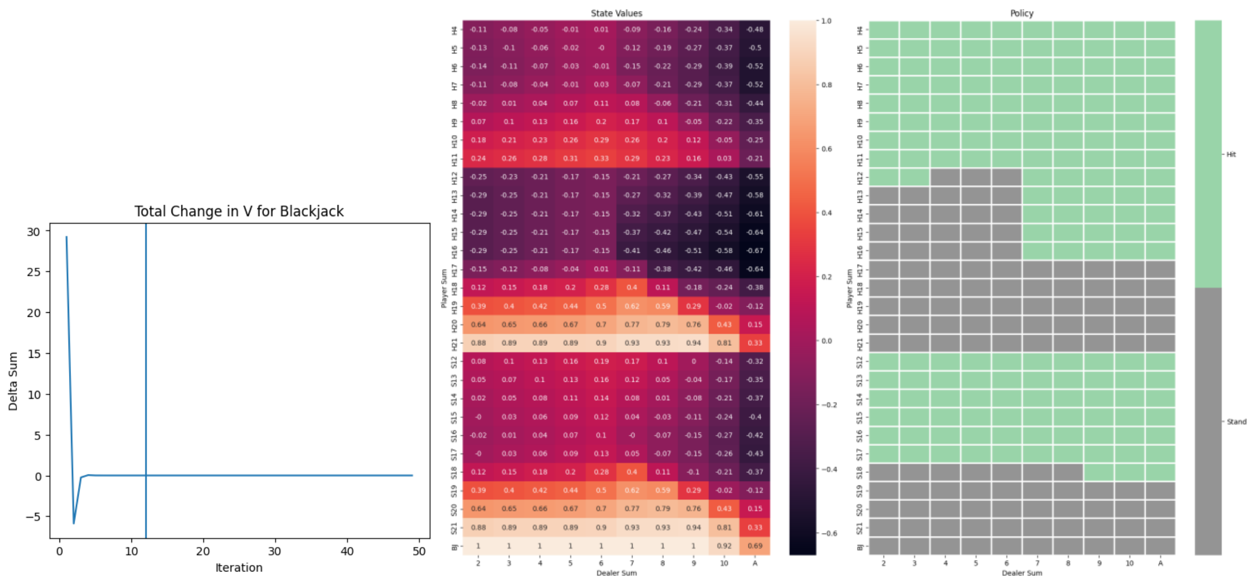

Figure 10. Total change in utility, Value of each state, Policy suggestion at each state after value iteration for blackjack MDP

The state value plot and policy suggestion plot have axes corresponding to the dealer card shown and the player card total. We see from the policy plot the optimal times to hit and the optimal times the optimal times to stand. From a casual player's stance, we see many trends that make sense, like how the player can be more aggressive with a usable ace, when otherwise, standing is more preferable. For instance, there exists a hard border in the middle of the policy suggestion plot, where hitting is best when the dealer's hand is at least 7, and player hand sum is a hard hand less than 16. Of course, the dealer only hits when their sum is lower than 17. If the player's hand is just under 17, then of course it does not want to stand knowing that the dealer will hit until over 17.

Value iteration converges at iteration 12 though the bulk of the value change occurs in the first three or so iterations. Compared to frozen lake, convergence is reached far earlier, and that is partly due to the smaller observation and action spaces of the blackjack MDP. Value iteration, which lets maximal rewards and the actions to reach them propagate through the observation space via the Bellman-Ford equation, more easily traverses the highly connected nature of the blackjack observation space, as any single state can be reached by numerous routes (there are several ways to reach a player state like H16). On the other hand, each state in the frozen lake problem can only even be reached by up to 4 states, via each of the actions, greatly increasing the number of iterations that are needed.

4.2. Policy Iteration

The total change in utility, state values, and policy are shown in figure 11 following policy iteration.

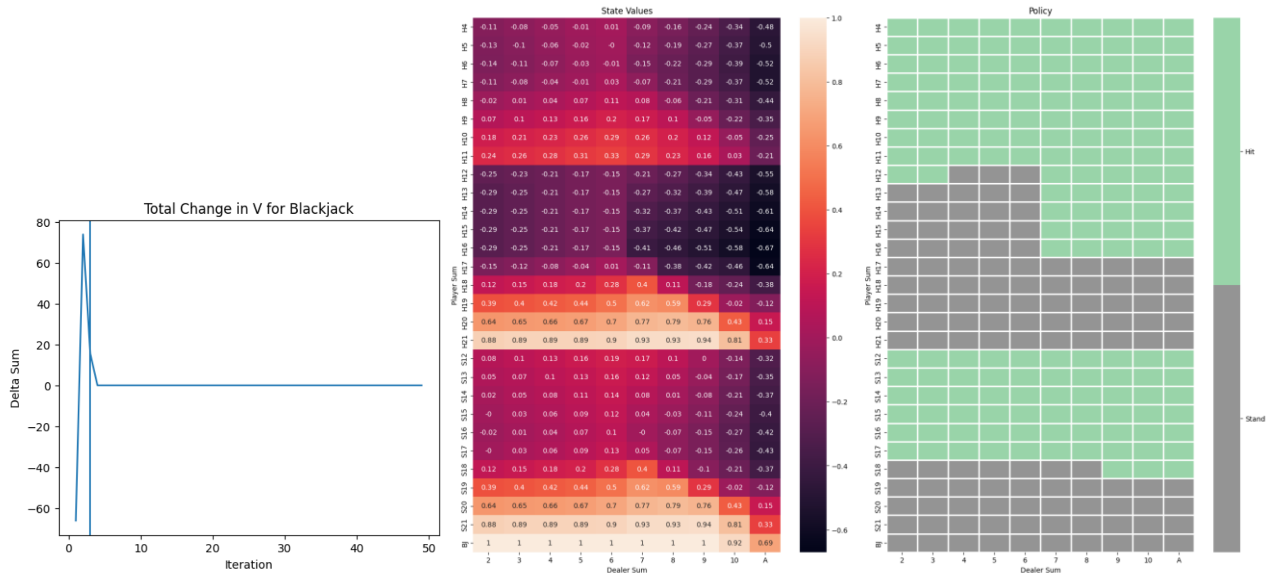

Figure 11. Total change in utility, Value of each state, Policy suggestion at teach state after policy iteration for blackjack MDP

Note that the policy suggestions of policy iterations are identical to those of value iterations. Though the optimal solution is not rigorously introduced, it would be probable that both policy iteration and value iteration converge on optimal solution, as per the policy suggestion plots. As with the frozen lake problem, the blackjack problem seems to converge in far less iterations with policy iteration than value iteration. The values at each state also seem to converge to identical values, as the total absolute error between the two state value plots total to only 2.808 * 10^{-14}. The only considerable differences between value iteration and policy iteration lie in the total changes in utility. Policy iteration first provides a significant dip in total change, which, is due to chance losses from busting given certain starting hands as well as double-hit-wins not being found. Surely, the total change rockets in the second iteration, as double-hit-wins are found, and chance busts are weeded out. Given another iteration, convergence is reached.

4.3. Q Learning

The state values and policy suggestions are shown in figure 12 following q learning. Q learning for blackjack was run with 1 million episodes. The total value change plot shows the sum change between episodes. We see that alpha is quite important in the decrease in total change, as mentioned before. Again, the total change in state value decreases to nearly a fixed value, due to the set minimum learning rate for q learning.

We obtain similar values from q learning than we do from past algorithms. However, there are a few glaring inaccuracies. The policy suggestion plot obtained from q learning seems to have a number of holes in an otherwise solid block of "hit", namely at H12. We also see a few states that q learning decides hitting is best, where other algorithms disagree, like S18. These are likely due to random error, as these errors are quite prevalent when examining earlier episodes of blackjack. With a larger number of episodes, we could see these inaccuracies disappear. Though we cannot say definitively which is correct, as both decisions have merit, it is worth mentioning.

The other large question is concerning the state value of the blackjack states, as seen in the final row of the state value table. Q learning suggests that the utility of these states is always exactly zero, not even negative. The other algorithms both have utilities of mostly ones, which makes sense, as having an ace and a ten is always winning (at the very least tying).

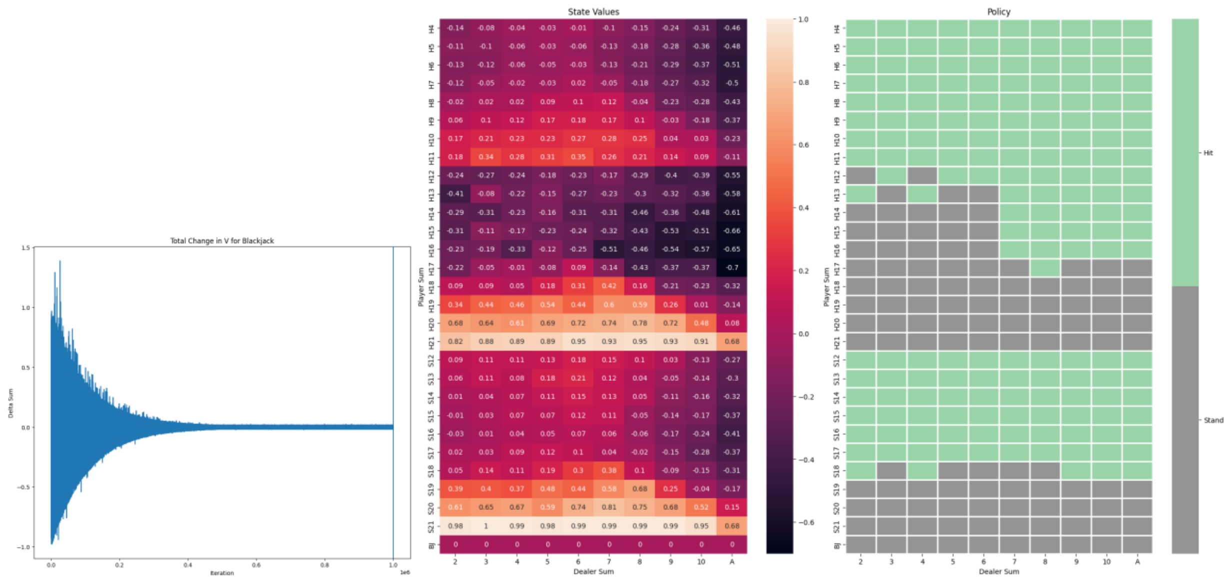

Figure 12. Value of each state, Policy suggestion at teach state after q learning for blackjack MDP

Regardless, we should see high utilities for in these states for q learning. One explanation for this phenomenon is the interpretation of these values. First, acknowledge that each element is initialized to zero before learning. Next, consider how q learning is performed. Q learning takes in a series of transitions, given a state, an action, a reward, and next state. However, what does a transition for this state look like? Because the blackjack hand is what is considered a terminal, meaning that no action is taken after because the game is finished, q learning will not find state-action-reward pairs that start with the blackjack hand. Therefore, the values are not updated from their initial state.

5. Conclusion

In this assignment, MDP's were solved through the lens of reinforcement learning. Three reinforcement learning techniques were considered: value iteration, policy iteration, and q learning. We saw that value iteration generally performed in more iterations than policy iteration, though each iteration took longer. Value iteration and policy iteration, both model-based algorithms, converge on near identical solutions for both frozen lake and blackjack. Q learning took a model free approach to solving the MDP's, meaning no model of transition matrices or reward structure was directly given before solving. We saw q learning take a directed route to finding the goal in frozen lake, and find a near-optimal solution for blackjack.