Assignment 4 : CS 7641

Muhammad Haris Masood
mmasood30@gatech.edu

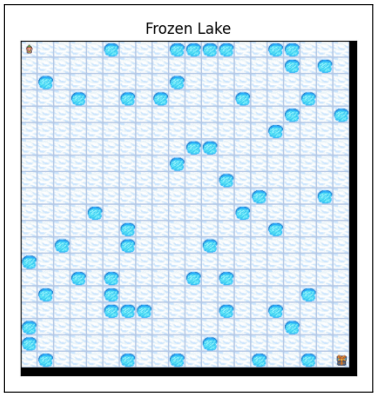

Figure 1. FrozenLake problem space

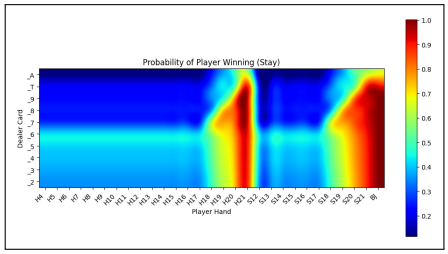

Figure 2. BlackJack problem space player win probability if stayed

1. Introduction and Problem Details

1.1. Overview

Figure 1 showcases the state space for our first Markov Decision Process (MDP) problem, FrozenLake, note the presence of the goal at the bottom right, the prevalence of holes and the generally complicated maze like structure of the problem. Figure 2 displays the state space of our second MDP problem, BlackJack, the figure helps visualize the win probability of a players hand if they choose to stay at that hand. Important features to note are the presence of several "goal/ideal states", and, the less traditional MDP structure as opposed to FrozenLake.

We chose FrozenLake as our large problem (20×20) due to its intricate structure, further compounded by the introduction of stochasticity in the form of slipperiness. In addition to its complexity, Frozen Lake presented a more conventional MDP problem structure, featuring one winning state and several losing states. Moreover, Frozen Lake facilitated easy visualization of the state space, enabling us to not only comprehend the problem but also gain an understanding of how various learning strategies, tuned with different hyperparameter values, approached the problem. In addition to these reasons FrozenLake offered easy scalability allowing us to ascertain the impact of problem size on our algorithms.

In contract to FrozenLake, BlackJack offered a problem with no single explicit goal, but, many potential "ideal states". Additionally, BlackJack introduced an adversarial aspect, markedly different from the FrozenLake problem, which could potentially result in distinct learner and agent behavior. For these reasons, we chose BlackJack as our second, smaller MDP problem, as it provided the opportunity to observe distinctly unique learner and algorithmic behavior.

2. Hypothesis

1) We believe that policy iteration will require a similar amount of time and iterations to converge as value iteration for simple problems. However, for large problems policy iteration will converge significantly faster [2].

2) We believe that policy iteration and value iteration should converge to the same value matrix and optimal policy as both aim to solve an MDP problem using Bellman's optimality principle.

3) We believe that Q-learning will require significant training time and careful hyperparameter tuning to balance exploration and exploitation, especially in complex problems with many states [5].

3. Experimentation Methodology

3.1. MDP Customization

For our BlackJack problem, we utilized the standard MDP problem provided by the OpenAI Gym library. Additionally, we incorporated an explicit transition matrix into our problem to ensure compatibility with model based algorithms.

For our FrozenLake MDP problem, we implemented several customizations to ensure algorithmic compatibility and the generation of meaningful results. Initially, we removed the step limit imposed by the environment, allowing runs to continue indefinitely if necessary. Additionally, we introduced a penalty for stepping into a hole, and we augmented the reward for reaching the goal from +1 to +10. This adjustment aimed to facilitate easier interpretation and color coding of visualizations. Furthermore, we experimented with adding a small negative penalty for each step to encourage the agent to reach the goal more quickly. While this sometimes led to more directed agent performance, it often caused the agent to end up in a hole prematurely to avoid accruing negative rewards. As a result, we removed this penalty.

3.2. Metrics

For our experimentation, we utilized several metrics. Some of the obvious metrics included average reward per run, the number of iterations required for convergence, the time taken for convergence, and the win rate. In addition to these standard metrics, we employed some more specialized metrics. These included the number of steps taken to reach the goal , the mean value for the value matrix, V_{mean}, and the Euclidean distance to assess convergence criteria and to identify differences between two matrices.

3.3. Procedure

Our paper progressed through two main phases, each comprising of two sub-phases. In the first phase, we focused on exploring the BlackJack problem. The initial sub-phase involved applying model based methods to the problem. Throughout this sub-phase, we varied the discount factors for both algorithms, generating a comprehensive dataset that enabled us to compare not only the performance of the algorithms against each other but also within themselves, observing how different discount factors affected their behavior. The second sub-phase entailed applying a Q-learning algorithm to the problem. Similar to the first sub-phase, we experimented with various hyperparameters, carefully documenting their effects on the learning process. Furthermore, we compared and analyzed the results obtained from Q-learning against those of our model based methods. To ensure the reliability of our findings, we repeated each combination of algorithm and hyperparameter at least five times. Our second main phase involved applying the same steps as discussed above to the FrozenLake problem, in addition, we also varied the number of states for FrozenLake and noted the impacts to our algorithms. Similar to Phase one, we ensured that each step was repeated and averaged several times.

A key point to discuss for all algorithms in both phases was identifying convergence. For our model based methods we specified convergence by setting a threshold value for differences in value matrices of subsequent runs, which was set at $\theta = 1 \times 10^{-10}$. We experimented with this value and noted that further decreasing the threshold did not bring about significant changes to our algorithms' performance. Identifying convergence in Q-learning was more challenging. We experimented with setting threshold values for differences between value matrices, threshold values for the Q-matrix, and by creating a check that identified when the ideal policy matrix had not changed for a large number of runs. Unfortunately, none of these convergence criteria led to to reliable or repeatable results. We noted that during Q-learning, there were several instances where these matrices did not change for a large number of runs (e.g., 10^5 iterations for our FrozenLake problem) but then suddenly incurred large changes in a relatively short amount of runs. For this reason, we identified the best method to identify convergence was to allow the Q-learning algorithm to run for a very large number of iterations, and then visually inspect the V_{mean} to identify convergence. This method proved both accurate and repeatable. We selected $2.5 \cdot 10^6$ for our FrozenLake problem, and $1 \cdot 10^5$ for BlackJack.

Regarding experimentation, we must note that we did not use the same ranges of discount factors, thresholds, epsilon decays, or learning rates at each step. This decision ensured that our experimentation results were meaningful and could lead to sound conclusions and in-depth discussion and analysis. For example, we used discount factors ranging from 0.1 to 0.99 for all algorithms applied to our BlackJack problem. Given the simplicity of the problem and the abundance of rewards, all discount factors yielded meaningful results. However, for the FrozenLake problem, we significantly limited our range of discount factors to between 0.57 to 0.999 for most of our experiments. We found that discount factors below this range led to poor agent behavior and anomalous results. We attribute this to the significantly higher complexity of the FrozenLake MDP problem and the scarcity of rewards in the environment.

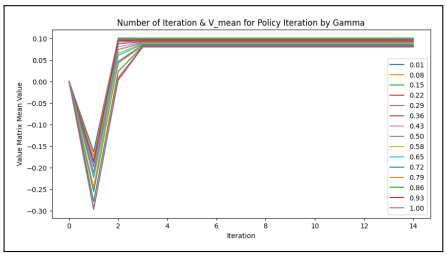

Figure 3. V_{mean} and iteration visualized for several discount values for policy iteration, BlackJack

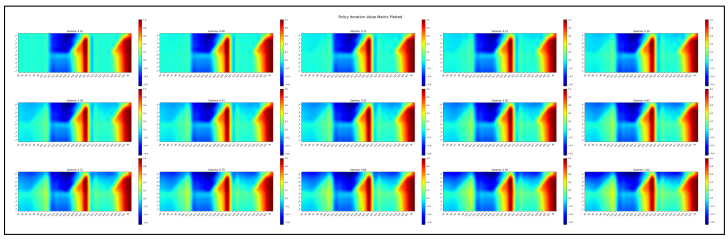

Figure 4. Value matrix visualized as a heatmap by gamma for policy iteration, BlackJack

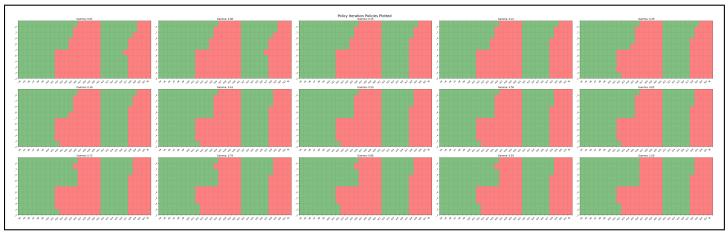

Figure 5. Ideal policy visualized for policy iteration color code is Hit and Stay, BlackJack

4. BlackJack

4.1. Policy Iteration

We began addressing the BlackJack problem by applying the policy iteration algorithm. Figure 3 illustrates V_{mean} for different gamma values across iterations. From our results, we observe that all policy iteration algorithms converged rapidly within a few iterations. Additionally, smaller gamma values resulted in faster convergence and a higher V_{mean} upon convergence. We hypothesis that small gamma values led to faster convergence as future states $\mathcal{N}$ transitions away had less of an effect on the value of the current state. Additionally, we believe this also explains the relationship between lower gamma resulting in the higher convergence value of V_{mean}. Consider the agent in state H16. The agent can either hit or stay. With a low gamma value, only immediate rewards matter. If the agent chooses to stay and wins, they receive a reward of +1. However, if the agent hits (and doesn't bust), they move to another state and receive a reward of 0. In the next state, if the agent hits again, they may bust, but the negative reward from busting doesn't flow back to the previous state due to the low discount factor. If the discount factor was higher, the negative reward would flow back and impact the value of the original state

Figures 4 and 5 help visualize the value matrix and ideal policy for different values of gamma in policy iteration. These figures allow us to draw some interesting conclusions. We observe that at low gamma values, the value matrix exhibits sharp, distinct areas where states representing high sums are considered high-value

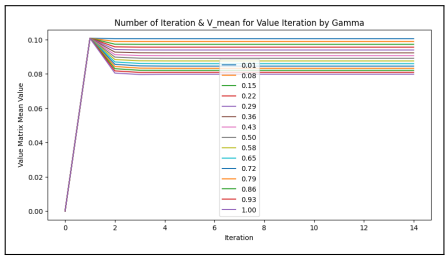

Figure 6. V_{mean} and iteration visualized for several discount values for value iteration, BlackJack

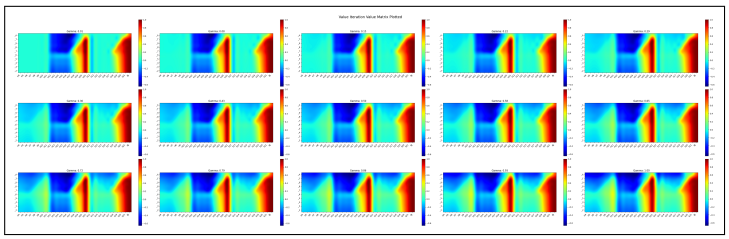

Figure 7. Value matrix visualized as a heatmap by gamma for value iteration, BlackJack

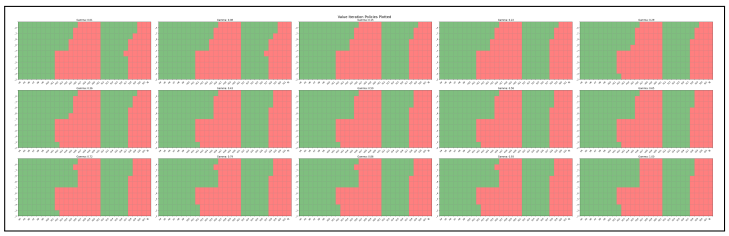

Figure 8. Ideal policy visualized for value iteration color code is Hit and Stay, BlackJack

states, while other states are either negative or zero value. This interpretation may not be accurate for anyone familiar with BlackJack, as low sum states can also have high value due to the player's ability to choose to hit. Unfortunately, a low gamma value fails to capture this nuance. This complexity extends to many other states, including hard and soft moderate sum states, when the dealer has a high card, and so on. Increasing the gamma value allows for more information to flow back to previous states, resulting in a more complex value matrix. This provides a richer understanding of the state space by accounting for potential future scenarios.

Turning our attention to Figures 5, we observe that low gamma values lead to a more risk averse policy. For example, when the dealer has a moderately high card such as 9 and the agent has a hard sum of 16, the agent does not hit at low gamma values. In contrast, at high gamma values, the agent chooses to hit. This indicates that with low gamma values, the agent is less willing to take risks, opting not to hit in order to avoid busting. However, a high gamma value may encourage the agent to take a risk, recognizing that a successful hit could lead to a significantly higher value state. This ties in with our initial discussion, a high gamma value allows the agent to have a better understanding of future states. We do note from Figure 15 that a riskier strategy only slightly improved win/reward rate, we do note that it does appear that a riskier strategy does encourage the agent to take more hits, as can been in average reward rate subplot in the same Figure.

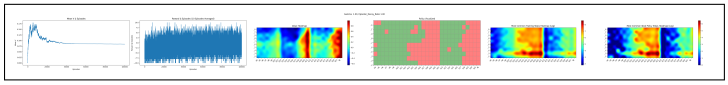

Figure 9. Ideal Q-Learning (gamma at 0.999, epsilon constant at 1.0, learning rate decays from 0.5-0.001), BlackJack

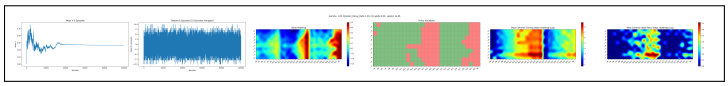

Figure 10. Q-Learning (gamma at 0.999, epsilon decay at 0.999, initial and final epsilon almost 0, learning rate decays from 0.5-0.001), BlackJack

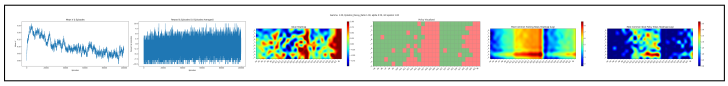

Figure 11. Q-Learning (gamma at 0.999, epsilon decay at 0.999, initial epsilon at 1.0, learning rate decays from 0.5-0.5), BlackJack

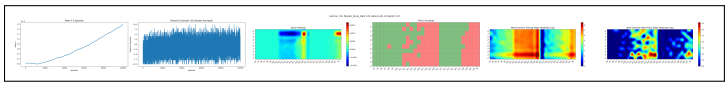

Figure 12. Q-Learning (gamma at 0.999, epsilon decay at 0.999, initial epsilon at 1.0, learning rate constant at 0.001), BlackJack

4.2. Value Iteration

Figures 6 visualizes V_{mean} for a several gamma values. We note that once again convergence takes place in a relatively few number of iterations, and, it does again appear that a higher gamma value leads to a lower V_{mean} upon convergence. We believe that the reasoning for this is similar to what we discussed in section 4.1. Interestingly, prior to convergence, the V_{mean} curve for value iteration differs from that for policy iteration. This could be due to value iteration operating in a more"greedy" manner, as it tries to maximize the value of each state at each iteration as opposed to evaluating the expected reward of the whole policy as done by policy iteration.

For value iteration the value matrix heatmap is visualized in Figure 7, and the ideal policy grid is visualized in Figure 8. We note that both methods produced near-identical value matrices and optimal policies for the same gamma. Value iteration's graphs followed trends similar to those observed in policy iteration, suggesting comparable underlying reasoning. A higher gamma value led to a richer value matrix and resulted in a riskier policy that encouraged more hits but only slightly improved win/reward rate (Figure 15). This gives credence to our initial hypothesis.

Despite producing similar results, both model based algorithms have some key differences in performance, which are discussed further in section 4.5 of the paper.

4.3. Q-Learning

For Q-Learning we began by varying several hyperparameters to note their effect on our learner. Our first step was to identify the ideal initial and final epsilon values, additionally we wished to note the effect of tampering with the exploration vs exploitation trade off. Figure's 9 and 10 help visualize this effect. In our experimentation we found that using a small value for epsilon led to slightly faster convergence (iterations and wall clock time), visualized by the V_{mean} subplots. However, the convergence V_{mean} values differed because low exploration resulted in a poorer understanding of the

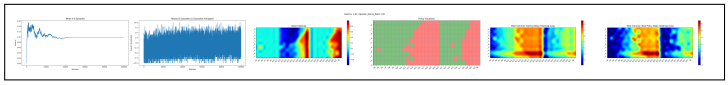

Figure 13. Q-Learning (gamma at 0.01, epsilon decay at 0.999, initial epsilon at 1.0, learning rate decays from 0.5-0.001), BlackJack

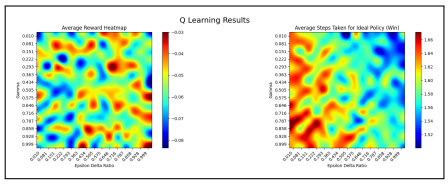

Figure 14. Results for differing gamma and epsilon decay values on Q-Learning, BlackJack

entire state space. This is most evident when comparing the most common training steps heatmap: lower epsilon values cause the agent to spend very little time in lower hard value states, leading to a poorer understanding of the value of these states. This lack of experience is visualized quite nicely in the ideal policy states heatmap. Unsurprisingly, low epsilon values resulted in poor performance, our testing indicated a 1%-3% reduction in win rate and on average a 0.15 lower reward per episode just by decreasing epsilon to a low value.

Our next hyperparameter was the ideal learning rate. From our experimentation we noted that a high learning rate resulted in erratic agent behaviour and a lack of convergence. Additionally, the resultant value matrix could be described as "overfit", these results have been visualized in Figure 11. We believe these issues can be traced back to erratic changes in the Q matrix brought about by the high learning rate. These erratic changes likely result in the lack of convergence, think a learner jumping around in a convex valley due to the high learning rate causing it to overshoot the bottom. The "over fitted" value matrix also indicates the high learning rate led to a poor understanding of the problem state space.These issues were particularly evident in the average reward: learners trained with learning rates higher than 0.1 experienced a significant decrease in performance, with an average reward reduction of 0.1.

Figure 12 shows a learner trained with a very low learning rate. Note the significantly slower convergence; in fact, convergence had not been achieved even after $1 \cdot 10^5$ steps. To ensure a fair comparison, we conducted experiments with low learning rates and increased the number of iterations to $2.5 \cdot 10^6$. While learners with a lower learning rate did eventually converge the benefits were not significant. For these reasons, we concluded that the significant increase in computation time and number of iterations was not worthwhile. Based on our experiments, we identified a decaying learning rate from 0.5 to 0.001 as the ideal learning rate.

Our final two hyperparameters were the discount rate and the epsilon decay rate. Rapid epsilon decay resulted in fewer observations, but, the many iterations per learner minimized the impact. We also saw no significant change in wall clock time or iterations for convergence. We believe this is due to Blackjack being a relatively simple problem; as long as the agent performs a reasonable number of initial exploratory steps, it can achieve a good understanding of the state space.

Lower discount rate did result in a larger effect as can be visualized in Figure 13. We note that low gamma values resulted in slightly faster convergence, both in iterations and wall clock

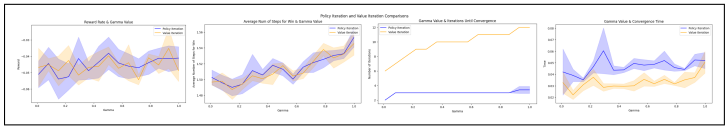

Figure 15. Comparison between value iteration and policy iteration for several metrics, BlackJack

time, but resulted in a significantly simpler value matrix. We believe the reasoning is similar to what we discussed in section 4.1. Interestingly small gamma values did not lead to any discernible degradation in average reward. This is further confirmed by our grid search visualizations, Figure 14. Note how there is no clear trend between gamma values and model performance. The simplicity of the problem and the lack of sparse rewards likely explain why low gamma values were still able to perform well. In blackjack, games end quickly in a few steps. We observed a slight trend between gamma values and the likelihood of the agent choosing to hit, which aligns with our our earlier discussion on the matter, in our model based sections. Faster epsilon decay also seemed to result in the agent requesting more hits. A potential hypothesis is that limited exploration has caused the agent to adopt a more rigid strategy, insisting on escaping from states with a low value sum (asking for hits) without considering the specific nuances of each state.

After hyperparameter tuning, the ideal Q learner achieved an average score of -0.03, which is comparable to the optimal policies generated by our model based methods. This demonstrates that, at least for smaller MDP problems, Q-learning is an effective strategy that can provide performance similar to model-based algorithms with access to the transition matrix. This finding supports the claim that Q-learning is an effective strategy for real-world problems.

4.4. State Size Variation

Unfortunately, the Blackjack problem we're experimenting with has a fixed state size, so we can't manipulate it to observe effects on our algorithms. However, existing literature and underlying mathematics allow us to formulate a hypothesis. Value iteration evaluates the value of each state and the optimal action for each subsequent state, leading to a time complexity of $O(S^2 \cdot A)$s [4]. Policy iteration approaches updates differently, incurring a time complexity of $O(S^2)$ while evaluating values, and $O(S^2 \cdot A)$s [4] while improving the policy. Resultantly we conclude that if we only alter state space, we should see a quadratic increase in time for our model based method.

Existing literature suggests that the increase in time complexity for Q-learning may be linear [3], but our experimentation shows that Q-learning is a sensitive algorithm significantly affected by the choice of hyperparameters and the problem itself. We believe this linear complexity should be considered a lower bound.

4.5. Comparisons between Algorithms

Our experimentation with the BlackJack problem resulted in us exploring our three algorithms in-depth. In this section we'll highlight some of the main differences we observed. We first look at the two model based strategies, results of which are highlighted in Figure 15. Both policy iteration and value iteration produced similar value matrices and policies, as well as, similar performance. However, there were differences in the number of iterations required for convergence, and to a lesser extent, the time taken to

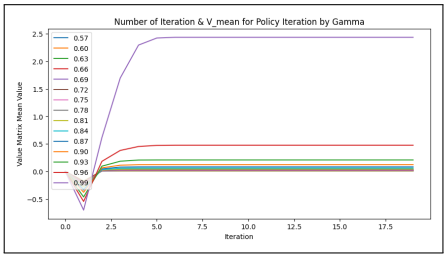

Figure 16. V_{mean} and iteration visualized for several discount values for policy iteration, FrozenLake

converge. Value iteration took significantly longer to converge because it is a straightforward algorithm that evaluates the value of each action in each state. This process requires more iterations to reach convergence. Policy iteration, on the other hand, uses a more efficient approach. It starts by identifying a potential optimal policy at each iteration and then evaluates that policy, making improvements wherever it finds better actions. This approach leads to it converging in a smaller number of iterations.

Although value iteration required more iterations to converge, it still did so more quickly. This may be due to the lower computational complexity of each iteration in value iteration, as the policy evaluation and improvement steps in policy iteration can be computationally expensive. For significantly larger problems, policy iteration may be more efficient in terms of wall clock time due to significantly lower iterations.

In our model-free approach, Q-learning took significantly longer to converge than our model based methods, we noted an increase of roughly 10-20 times in both the number of iterations and wall clock time. However, performance was quite comparable to our model-based strategies, and Q-learning did not require prior knowledge of the transition matrix, which is almost never available in real-world and continuous problems.

5. Frozen Lake

5.1. Policy Iteration

In phase two, we tackled the FrozenLake problem. As with our approach to BlackJack, we began by applying policy iteration. Figure 16 showcases the V_{mean} for different values of gamma. Compared to our results for BlackJack we note that it takes more iterations to reach convergence and the V_{mean} value observed at convergence is higher. Additionally, the convergence V_{mean} value increase with discount factor. We believe these observations can be explained by the characteristics of the FrozenLake problem. FrozenLake is a significantly more complex MDP problem than Blackjack, so it's not surprising that more iterations are needed to reach convergence. The increase in V_{mean} with higher gamma is likely due to the sparsity of positive rewards in FrozenLake; the agent only receives a reward upon reaching the goal. Low discount values impede the reward from flowing back to past states, resulting in an overall lower value matrix. Finally, the reason V_{mean} converges at a higher value is because we adjusted the agent's rewards. The agent receives a reward of +10 upon reaching the goal and a penalty of -1 if it falls into a hole.

Figure 17 helps confirm our discussion on the effect of discount factor and V_{mean}. We note that with increasing gamma values the goal state "radiates" more of the positive reward to further away states.

Figure 18 presents the ideal policy for different gamma values.

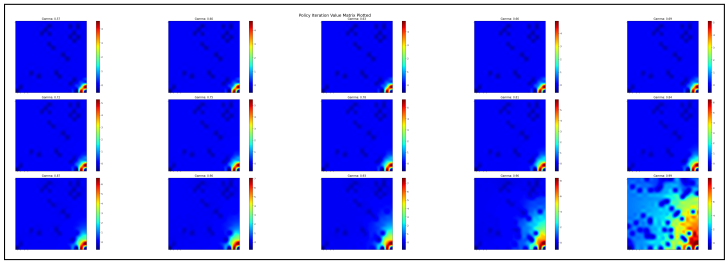

Figure 17. Value matrix visualized as a heatmap by gamma for policy iteration, FrozenLake

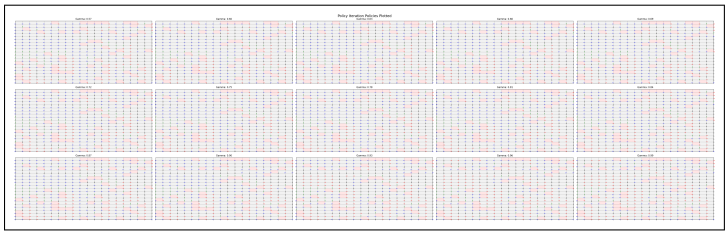

Figure 18. Ideal policy visualized for policy iteration color code is Holes, goal state is at the bottom right for each grid, FrozenLake

Figure 19. Ideal policy for policy iteration differences highlighted, FrozenLake

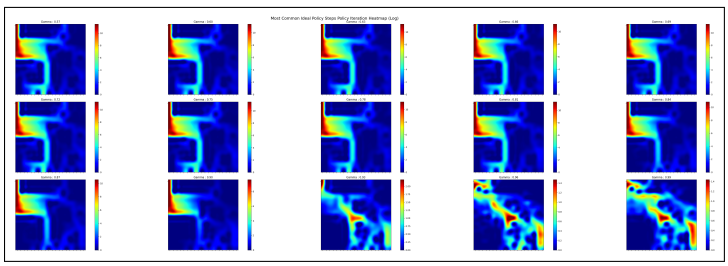

Figure 20. Most common steps of agent implementing ideal policy, policy iteration, FrozenLake

Notably, higher gamma values result in an optimal policy that is more motivated to reach the goal state, even if it means taking risks by approaching some holes. We can see these differences highlighted in Figure 19, note how the low gamma ideal policy tries to avoid the large cluster of holes found on the right of the lake, the high gamma policy is less risk averse. We believe the reasoning for this behavior is due to the assigned reward values and the sparsity of rewards in FrozenLake. With a low discount value, the reward of reaching the goal state does not trickle down to states further away; instead, the policy is more influenced by nearby holes. Although the holes themselves carry a significantly smaller negative reward than the positive reward offered by reaching the goal, it is still substantial enough to motivate the policy to avoid them. This issue is resolved with higher gamma values, which allow the reward to be propagated across a larger portion of the lake.

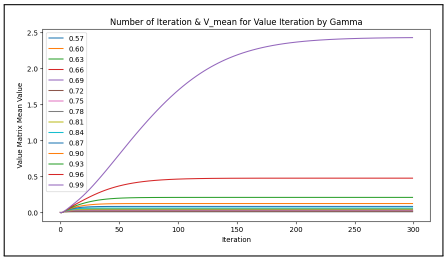

Figure 21. V_{mean} and iteration visualized for several discount values for value iteration, FrozenLake

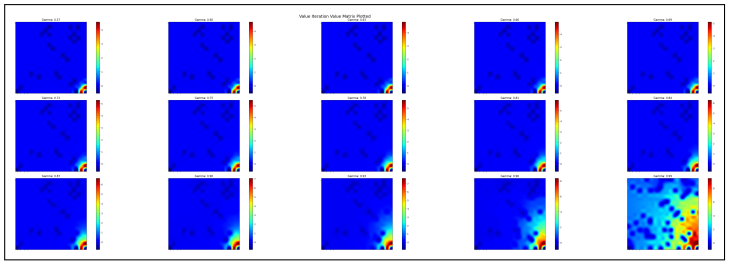

Figure 22. Value matrix visualized as a heatmap by gamma for value iteration, FrozenLake

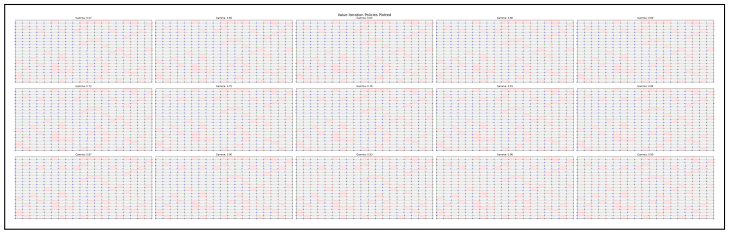

Figure 23. Ideal policy visualized for value iteration color code is Holes, goal state is at the bottom right for each grid, FrozenLake

Interestingly we note that this more cautious policy/agent behaviour did not result in more frequent wins/greater average rewards (Figure 34) if anything it appeared to decrease the win percentage. We believe this is due to the surface being "slippery". The more time the agent spends on the lake, the higher the likelihood that stochastic processes will result in it ending up in a hole. Conversely, we note that a higher gamma value ("riskier") resulted in better average scores/wins. We do note that the more cautious approach does result in significantly more average number of steps to reach the goal state. This can be explained by observing the state space. The direct most path to the reward straddles closely with several holes, the agent likely takes the "long" way around to avoid this high risk zone. In fact, we can confirm this by examining Figure 20, which visualizes the steps taken by our agent across a large number of runs. We observe that for high values of gamma, the agent adopts a much more direct path.

5.2. Value Iteration

Figures 21 visualizes V_{mean} for a several gamma values for value iteration performed on our FrozenLake problem. Here again, we observe that as the discount rate increases, the converged V_{mean} value also increases. We believe this is due to the same reasoning as discussed in section 5.1: the sparsity of rewards, the large state space, and the lack of reward flow back to earlier states unless there is a high gamma value.

Figure 22 helps us visualize the optimal value matrix for different gammas, and again, we note that a high gamma is needed to radiate the reward sufficiently throughout the state space. The effects of poor reward propagation can be viewed through Figure's 23 and 24. If the discount rate is too low the agent is not driven to end up in the reward state, instead, it's focused more on avoiding the holes and this translates to generally poor performance as observed in Figure 34. Stay too long on the ice you'll end up in a hole eventually!

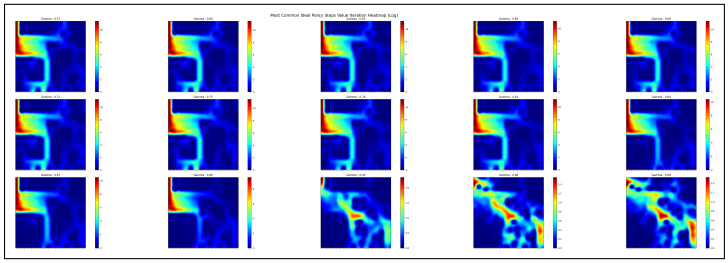

Figure 24. Most common steps of an agent implementing ideal policy, value iteration, FrozenLake

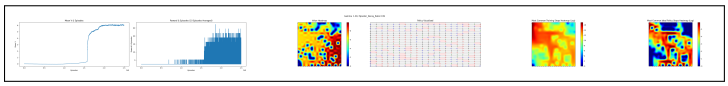

Figure 25. Ideal Q-Learning (gamma at 0.999, epsilon decay at 0.999, initial epsilon at 1.0, learning rate decays from 0.5-0.1), FrozenLake

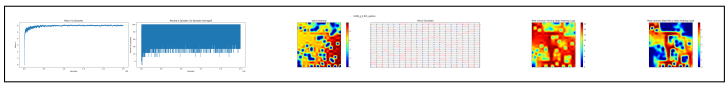

Figure 26. Q-Learning (gamma at 0.999, epsilon decay at 0.999, epsilon constant at 0.001, learning rate decays from 0.5-0.1), FrozenLake

For the FrozenLake environment, the optimal policy derived from value iteration differed from the optimal policy derived from policy iteration, even though the value matrices were identical. This is a notable difference between FrozenLake and Blackjack. The differences in policies were minimal; the largest discrepancy occurred at a discount rate (gamma) of 0.63, where the optimal policy differed by 6 steps between our methods. Initially, this result was puzzling, but an in-depth literature review revealed that there can be multiple optimal policies [1].

5.3. Q-Learning

Similar to BlackJack we began applying our model free approach of Q-Learning by first performing some extensive hyperparameter tuning. We began by identifying the effect of varying epsilon. We can note the effect by comparing Figures 25 and Figure 26. A lower epsilon value lead to less exploration and caused the agent to be more rigid in the paths it took to reach the reward. This can be observed in the heatmaps of training steps and ideal policy steps. Note the stark difference in colors for the exploration map with a low epsilon value, indicating limited exploration. In the ideal policy steps heatmap, the effect of this can be seen in the significantly fewer number of routes taken to the reward. Surprisingly it does appear that even a very low epsilon value led to strong performance, in-fact, the average reward and number of steps taken to arrive at the goal were quite comparable. Additionally, as can be seen from the convergence plot, a learner which focused heavily on exploration converged much faster both in wall clock time and number of iteration. The question thus arises: "Why not use a low

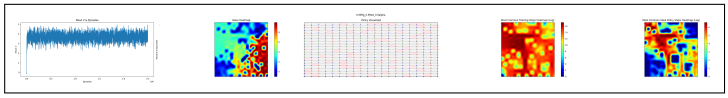

Figure 27. Q-Learning (gamma at 0.999, epsilon decay at 0.999, epsilon at 0.001, learning rate constant at 0.9), FrozenLake

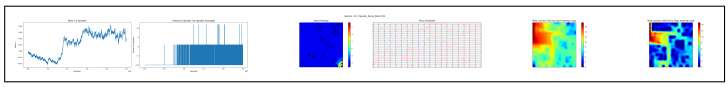

Figure 28. Q-Learning (gamma at 0.72, epsilon decay at 0.999, epsilon at 0.001, learning rate decays from 0.5-0.1), FrozenLake

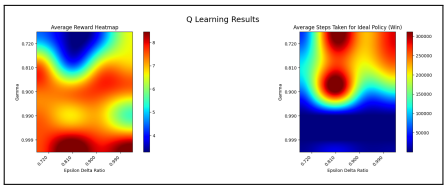

Figure 29. Results for differing gamma and epsilon decay values on Q-Learning, FrozenLake

epsilon value throughout?" The answer is that while a low epsilon value worked in this particular scenario, it performed poorly with lower discount values. In our experiments, even a discount value of 0.99 often failed to converge, or, resulted in extremely lengthy learning phases. We believe the low epsilon value prevented the agent from quickly finding the reward. Instead, the agent avoided holes, and when it eventually reached the reward through small random processes, the high discount value led to minimal positive reward at the initial steps, which occurred a large number of steps earlier. As a result, we found that using a decaying epsilon value from 1.0 to 0.1 was a good choice.

For our learner we also varied the learning rate and noted it's effect. Similar to what was observed for BlackJack too high of a learning rate lead to unstable convergence, as the V_{mean} and Q matrix had frequent large changes this can be easily observed in Figure 27. Due to this erratic behavior, we could not confirm if the algorithm had even converged, resultantly we are unable to comment on the iterations and time required to converge. Additionally, the unusual heatmap of the value matrix is a result of the algorithm's erratic behavior. In terms of performance, a high learning rate resulted in significantly poor outcomes. The learner in our Figure had a win rate of only 45.4%, which is half the rate of many other, more reasonably trained learners.

For very low learning rates, convergence took a long time in terms of both wall clock time and iterations and resulted in no significant improvement in terms of average reward. After experimenting with different learning rates, we found that a decayed learning rate from 0.5 to 0.1 was ideal, providing excellent performance and reasonable convergence time.

Our final two hyperparameters were the discount rate and epsilon decay. We noted that the epsilon decay value had a muted effect on learner performance. We believe this is likely due to the fact that we are performing a large number of iterations, allowing even a quickly decaying epsilon to result in adequate exploration.

Low gamma values led to markedly poor performance. Figure 28 illustrates this. The sparse value matrix heatmap indicates poor reward propagation. Additionally, the most common agent steps demonstrate a focus on avoiding holes rather than arriving at

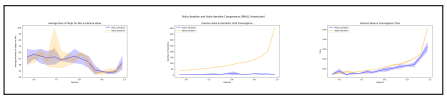

Figure 30. Various metrics measured for policy iteration and value iteration applied on a (5×5) FrozenLake

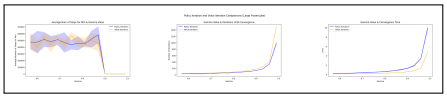

Figure 31. Various metrics measured for policy iteration and value iteration applied on a (25×25) FrozenLake

the reward state. We believe this low discount value results in a policy aimed more at avoiding holes than at reaching the reward. Interestingly, low discount rates required fewer iterations to converge but took more time overall. This was likely due to very long episodes where the agent moved back and forth in a safe space to avoid holes.

Figure 29 helps visualize the results of varying gamma and epsilon decay values. As expected we note better model performance when the gamma value is high (good radiation of reward state), and, the rate of decay of epsilon is small (good exploration). Additionally, we observed a similar trend as seen in policy and value iteration: low gamma values led the agent to take significantly more steps per episode. The worst combination appeared to be low gamma paired with rapid epsilon decay.

Our ideal Q learner achieved an average episode reward of 8.62, which corresponds to an 86.2% win rate. This is a notably impressive performance, comparable to our model based learners. Remarkably, the Q learner accomplished this without relying on a transition matrix, demonstrating its effectiveness even in challenging problem spaces such as FrozenLake.

5.4. State Size Variation

We took advantage of the FrozenLake environment's scalability to ascertain the effect of state size on our algorithms. We began by observing how varying state size impacts our model based methods. We present results for several metrics for a small FrozenLake in Figure 30 and a very large FrozenLake in Figure 31. By using these figures alongside our original value and policy iteration results (Figure 34), we can identify some interesting trends.

We note that the number of iterations required for value and policy iteration scaled logarithmically with state size. Focusing on value iteration, we observe that for a gamma value of 1.0, the algorithm converged in 400 iterations for a small state space, 800 for a large state space, and 1200 for a very large state space. However, the state sizes increased by a factor of $16\times$ going from small to large, and by $1.5\times$ going from large to very large. In fact, for lower gamma values, the differences in the number of iterations are even less proportional.

We believe the logarithmic increase in iterations required for convergence is due to the way we increased the problem size.We increased the number of states, but the complexity within each state remained the same, as the agent still had only four actions per state. Therefore, the iterations required for convergence have only grown logarithmically because the overall state complexity has not significantly increased. We suggest this is a potential area for future exploration, in which both state size and state complexity are increased by providing the agent with more actions per state.

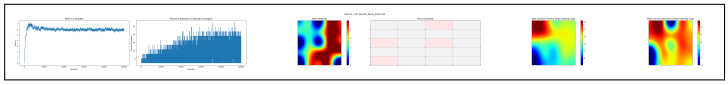

Figure 32. Q-Learning (gamma at 0.999, epsilon decay at 0.999, initial epsilon at 1.0, learning rate decays from 0.5-0.1), (5×5) FrozenLake

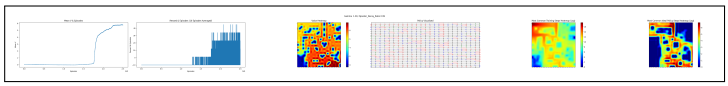

Figure 33. Q-Learning (gamma at 0.999, epsilon decay at 0.999, initial epsilon at 1.0, learning rate decays from 0.5-0.1), (25×25) FrozenLake

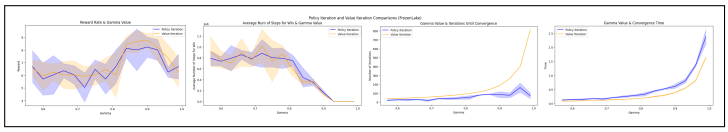

Figure 34. Comparison between value iteration and policy iteration for several metrics, FrozenLake

Turning to wall clock time required to converge we note polynomial increase, being in line with industry research. Our small FrozenLake at a gamma value of 1.0 converged $80\times$ faster for policy iteration than our large problem.

Interestingly it appears policy iteration converges faster than value iteration only for our small state. This is an interesting result and it refutes our initial proposition that an increase in state size may result in policy iteration converging faster than value iteration in terms of wall clock time [2].

We were also able to ascertain the impact of state size on Q-Learning. Figure's 32 and 33 showcase the results of strong Q learners on the small and very large FrozenLake problem. In the case of Q-learning, we observed that the number of iterations and wall-clock time required for convergence were more complex compared to our model-based methods. Due to the stochastic nature of states, the number of holes present, reward shaping, and learner hyperparameters, the time and iterations to converge could increase linearly with state size (Large vs Very Large FrozenLake) or even exponentially (Small vs Large FrozenLake). Our findings make sense, as Q-learning is a model-free approach that relies on a delicate balance between exploration and exploitation to solve the problem. Increasing state size likely has a complex relationship with this balance, resulting in unexpected changes and outputs.

We would like to note that for all three state sizes it did appear based on our findings that Q-Learning was a viable strategy, but, may require more finer reward shaping and hyperparameter tuning with larger problem sizes.

5.5. Comparisons between Algorithms

Utilizing Figure 34 we can compare our two model based methods for our FrozenLake problem. We note that in terms of performance both algorithms performed quite similarly, achieving the same reward rate for every gamma value. This comes as no surprise as our value matrices and optimal policies converged to nearly identical values. We do note that policy iteration took fewer iterations to converge, but, required longer wall clock time. Here again we believe the reasoning is identical to what we outlined in section 4.5.

Q-learning took significantly longer to converge compared to our model-based algorithms. We believe this is due to the complexity of the problem. Without a transition matrix to guide the agent, the agent must explore a large portion of the state space to generate an ideal policy. The stochastic nature of the problem further adds to this complexity. In terms of performance, we found that a properly tuned Q learner is able to achieve results that are comparable to those of our model-based methods. This indicates that Q-learning is an effective and practically applicable strategy in real-world scenarios.

6. Conclusion and Next Steps

Through our experimentation and analysis, we have deeply explored reinforcement learning. We believe we are now able to draw conclusions about our initial hypothesis. Our first hypothesis stated that policy iteration would converge faster, in terms of iterations and wall clock time, particularly for larger problems. However, we found that this might not be entirely accurate, and the reality appears to be more nuanced. Our experimentation with the Very Large FrozenLake environment showed that value iteration seemed to converge faster in terms of wall clock time compared to policy iteration. Furthermore, our mathematical analysis in section 4.4 suggests that the convergence time might be influenced by a complex set of factors.

Our second hypothesis was that policy iteration and value iteration would converge to the same optimal policy and value matrix. This was partially correct; both algorithms converged to the same value function. However, our FrozenLake experimentation revealed that the optimal policy was not necessarily the same for both algorithms. Upon reviewing the literature, we now believe our initial hypothesis was too narrow in scope. Both methods converge to an optimal policy, but a problem may have more than one optimal policy. This explains the observed differences in our FrozenLake experiment, where the problem likely had multiple optimal policies.

Finally, we hypothesised that Q-learning requires a significant amount of time to converge and is sensitive to hyperparameters and the state space itself. This was confirmed to be true during our experimentation. Even for a relatively simple problem, we observed that poor hyperparameter tuning greatly impacted the balance between exploration and exploitation, resulting in rapid degradation of the learner's performance.

Although our experimentation was detailed and in-depth, not all our questions were answered, and new ones arose from our results. The complex relationship between time complexity and state size for reinforcement algorithms, the scalability of learning algorithms to large problems, and the selection of the optimal policy from a range of possibilities are areas that need further exploration.

References

[1] McGill Faculty. Lecture 16: Markov decision processes. policies and value ... 6

[2] Mohand Hamadouche, Catherine Dezan, David Espes, and Kalinka Branco. Comparison of value iteration, policy iteration and q-learning for solving decision-making problems. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 101–110, 2021. 1, 8

[3] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I. Jordan. Is q-learning provably efficient?, 2018. 4

[4] Dan Klein and Pieter Abbeel. Uc berkeley cs188 intro to ai, 2019. 4

[5] Tiago Ribeiro, Fernando Gonçalves, Inês Garcia, Gil Lopes, and A. Fernando Ribeiro. Q-learning for autonomous mobile robot obstacle avoidance. In *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 1–7, 2019. 1