Reinforcement Learning Report

Skyler Ebelt
Georgia Institute of Technology
GT ID: 904077010

I. AI Use Statement

In order to complete this project, AI was utilized in a few ways. I only used Claude Sonnet 4.5 for the duration of this project. Claude was utilized for organization, to better manage my workflow on this project. Additionally, Claude was used to refactor broken code segments. I worked to understand every change suggested by Claude before implementing it in my code.

II. Introduction

A. Enviroments for Analysis

For this final assignment, we finally break free from the curse of the Hotel and Accident datasets which were previously used. This assignment utilizes two different types of Markov Decision Processes (MDP). By way of the package gymnasium, a Blackjack MDP [1] as well as a physics based CartPole MDP [2] were established for the purposes of this assignment.

The blackjack MDP represents a turn based card game in which the player attempts to obtain a score higher than that of the dealer. Decisons in this specific circumstance is limited as compared to its real world counterpart, with the player only being able to hit (draw an extra card) or stand (hold position and wait for dealer to reveal card). Traditionally blackjack also allows splitting (when given two of the same card, split into two hands and receive one more card on each, then decide to hit or stick for each new hand) or to double (double down wager, can double with any hand before drawing next card, though standard blackjack strategy will point to only a few explicit scenarios where this is appropriate. This a discrete problem, as it focuses on the total score of the player and dealer. Actions, again, are a binary space, hit or stand (0,1). Card draws are stochastic, card draws follow a probabilistic distribution. The episode will be terminated when player stands or busts. The dealer netting a score below the player nets a win for the player, the dealer busting nets a win for the player. Conversely, the player with a score below that of the dealer nets a loss for the player, and the player busting nets a loss for the player.

This environment provides an incredibly interesting decision space for a learner to navigate. Due to the probabilistic nature of blackjack, but also the capacity for some strategy provides a circumstance which will force the learner to balance risk. The learner figuring out the nature of the game may develop strategies to best balance risk to reward.

The CartPole MDP simulates a physics environment in which a cart contains a pole. The agent must move the cart along a track while balancing the pole. This environment provides a 4D vector which contains the cart position, velocity as well as the pole angle and pole angular velocity. The agent can once again make decisions within a binary space, push cart left or right (0,1). The agent is rewarded for each timestamp for which the pole is upright. End conditions are determined by if the pole falls, or if 500 timesteps are achieved. Discretization is required for this implementation. Representing a continuous space as a discrete space is necessary in ensuring the agent can interpret cart and pole states. Two schemes have been selected, a small scheme with 4 bins per state, and a medium scheme with 6 bins per state.

This presents a continuous problem with deterministic dynamics. This contrasts blackjack which contains a discrete issue with probabilistic dynamics. This should highlight difference in algorithmic performance in different problem spaces. This problem should allow the model to learn and predict how each action translates to the poles position, perfectly learning the problem. Due to the use of discretization however, there is a large tradeoff of granularity for computational cost.

B. Hypothesis

For the purposes of this assignment the following hypothesis has been derived: Agents will net better performance and faster convergence within the deterministic CartPole environment as opposed to the stochastic blackjack environment, we expect to see models better understanding the deterministic dynamics. The main difference between the two environments is the problem domains. Due to blackjack having a component of agent "luck" or in other words, entropy in what card a hit would yield, it should be expected that these agents will not perform as well. The nature of the deterministic dynamics should enable agents to perfectly predict the effect each action has, whereas an agent cannot perfectly predict what card will be drawn. This uncertainty is going to limit the capabilities of the agents, while they may still perform well, we expect the agents to not be able to completely and fully solve the problem. If this were possible, online casinos/gambling sites would likely no longer exist. This hypothesis directly relates to the experiments which will be discussed further and should be directly supported or not supported in the results and discussion of each experiment.

III. Environment Setup & Hardware

A. Environments

The blackjack environment was established with sab = True. This ensures the agent observes all state components. Next natural = False was used to simplify the structure for rewarding the agent. This set the values of +1, 0, -1, for a win, stand, and loss respectively. Because the issue of blackjack is already a discrete problem, no discretization was utilized to establish this testing environment. 5,000 episodes were utilized to ensure the agent could explore all states and the decisions associated with them. This process yields 280 unique states. The probabalistic nature of the blackjack environment means that each state has a number of possible states that can follow them due to the random nature of drawing additional cards.

Establishing the CartPole environment required a much different approach due to the contrasting nature of the environment to that of the blackjack environment. This problem space required discretization due to the continuous nature of the physics based data being fed to the agent. This requires feeding the agent an output of the current state by way of bins. This allows the agent to see what is happening within the simulation incrementally at each timestep, enabling its operation. Each episode is limited to 500 timesteps, with a success threshold of >= 475 timesteps passed. The two seperate schemes were afforded different budgets when running. The small scheme was allowed 3,000 episodes, and the medium scheme was allowed 5,000 episodes to ensure better coverage over the larger state space

Planning methods were given a discount factor of 0.99, a convergence threshold of 1e-10 and a max iteration of 1,000. Learning methods were given a learning rate of 0.1, epsilon of 1, epsilon decay of max(0.01 * 0.9995), a discount factor of 0.99 and 50,000 episodes for blackjack and 5,000 for CartPole. For reproducibility, I broke my traditional scheme and selected a seed of 42.

B. Evaluation & Hardware

For executing these experiments within these environments, my PC contains 32 GB of ram. This is paired an i7-9700. Due to the nature of these experiments, utilizing no datasets, no modifications have been made to any samples as this was not a requirement for this assignment. All experiments executed in a timely manner, no computational constraints were violated in these experiments.

IV. Methodology

A. Value Iteration

Value iteration is an algorithm which computes optimal value functions through iterative Bellman updates. The algorithm assumes that the value of its state equals the max expected return which can be derived from that state. This initialization of this algorithm requires initializing V(s) = 0 for each state. Next the Bellman equation is used to calculate Q(s,a). From here each instance of V is updated as the max value of Q. This method guarantees convergence to the best policy. Sweeps are performed over the state space within each iteration.

B. Policy Iteration

Policy iteration alternates between evaluating the policy, but also policy improvement. This allows agents to explore while attempting to optimize the policy. This can potentially allow for faster convergence. On average we should expect this policy to converge faster than Value Iteration, however each iteration becomes more expensive due to the policy evaluation. This means larger computational costs for an algorithm which should work faster.

C. SARSA

This model free algorithm learns action values by updating based on the action taken by the behavior policy. This algorithm reads a state, takes an action, reads a reward, considers a next state and finally considers a next action. Decision making is allowed to be exploratory with an epsilon greedy policy being used within this instance. Here, an exponential decay is used to ensure the algorithm can explore decision space, but with each iteration it should progressively begin to follow the optimal policy. By allowing for exploration, it is hoped that the model will directly learn the optimal policy within the allotted episodes. This method is more conservative than Q-learning because of this, as it learns about the policy it is actively following.

D. Q-Learning

Q-Learning, while similar to SARSA, differs in that it is a off policy learning method. This means it will take the optimal action value regardless of the followed policy. This algorithm makes decisions by calculating what the optimal next value is of the following state. This algorithm will observe its state, choose an action using epsilon-greedy, observe its reward, and then observe its next state. This method differs from SARSA in that it will learn the policy it is following, while exploring around it. This allows it to learn from mistakes, while still converging to the optimal value.

V. Results and Analysis

A. Blackjack Results

Both value iteration (VI) and policy iteration (PI) converged rapidly. In the case of VI, 10 iterations were needed for convergence, and PI required just 4 iterations to converge. This should be expected as PI can stabilize policy without needing value estimates to converge. Both algorithms required the same time to run with 0.03 seconds, showing that while PI converged faster when considering iterations, its tendency to perform multiple sweeps kept it comparable to VI within runtime.

Both VI and PI netted rewards near 0 suggesting this environment allowed them to generate effective strategies for decision making. However, this does suggest decision making was conservative. This was likely required due to

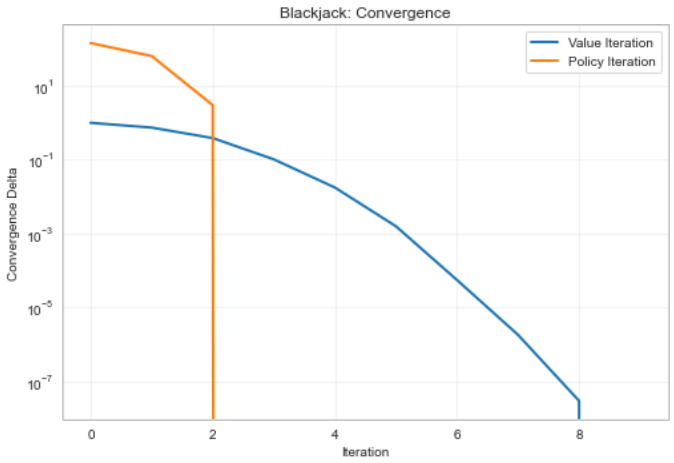

Fig. 1. Blackjack Convergence Plot

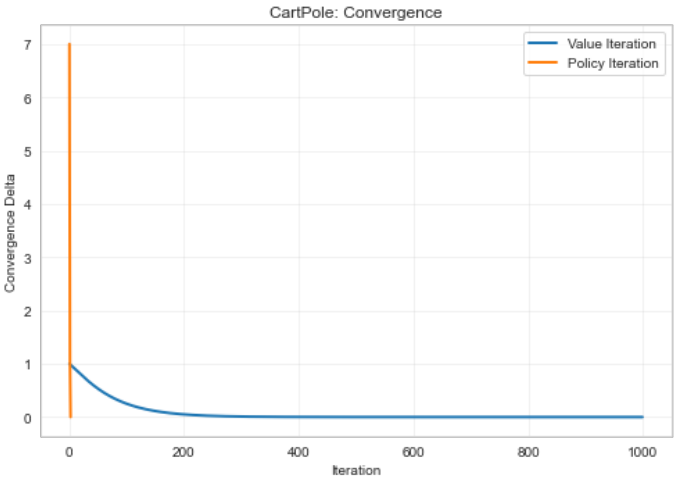

Fig. 2. CartPole Convergence Plot

this problem being stochastic, in that it was dealing with card draws.

SARSA and Q-Learning yielded similar results to VI and PI, but required significantly more time to train. All models netted a mean reward which was negative, despite being near zero. This can be considered evidence to support our hypothesis. If agents were able to perfectly solve this issue, we should expect positive rewards. Because this issue is stochastic and not deterministic, the model cannot anticipate how to perfectly beat the dealer. In addition, there are many circumstances in which a hit will net a bust for the agent, but a stand will cause the dealer to beat the agent. In other words, we expect states to exist in which, regardless of the decision taken by the agent, it will lose no matter what. This negative, near zero mean reward is evidence of this phenomenon and supports the initial hypothesis which suggested this would be the case.

High variance can be observed across all learners within the blackjack environment. This further asserts that the randomness, and especially cases of total defeat (any action taken by agent yields a loss) makes this unsolvable, at least

TABLE I

BLACKJACK PERFORMANCE COMPARISON ACROSS ALGORITHMS

Algorithm	Mean Reward	Std Dev	Training Time
Value Iteration	-0.007	0.959	0.03s
Policy Iteration	-0.061	0.956	0.03s
SARSA	-0.048	0.960	5.91s
Q-Learning	-0.075	0.947	6.09s

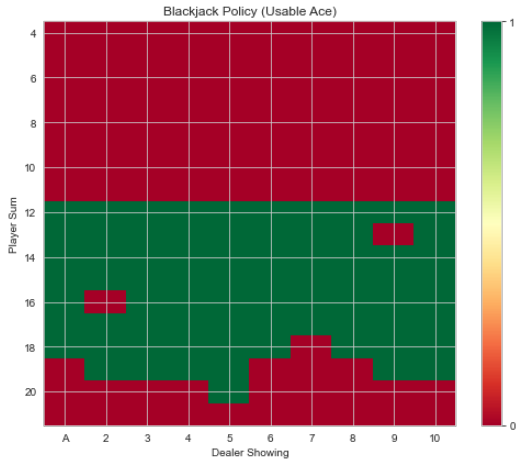

Fig. 3. Blackjack Heatmap: Usable Ace

to "perfection" by any agent. This does not suggest these learners are ineffective, but rather that while deriving optimal policies, these policies cannot perfectly solve this problem.

Despite all models performing similarly, we can observe a massive advantage to the PI/VI methods. Time to train was a fraction of that of the model free based agents. This can be advantageous for much more extensive and complicated problems for agents to learn, showcasing that when pushed to their limits, their train time would remain much lower than that of the model-free based methods.

Considering learning curves of both SARSA and Q-Learning contrast the two methods. Both methods improve over the allotted episodes with significant noise due to the stochastic nature of the blackjack environment. Both converge to similar performance suggesting well suited parameters for the learners.

B. CartPole Results

Within the CartPole environment, drastically different behavior was exhibited when considering convergence by each agent. VI within both discretization levels failed to properly converge. PI was only able to converge within the small discretization level. This indicates issues with either implementation or within the transition model. When PI was able to converge however, it required only 3 iterations to do so.

TABLE II

CARTPOLE PERFORMANCE WITH SMALL DISCRETIZATION

Algorithm	Mean Reward	Std Dev
Value Iteration	19.6	19.6
Policy Iteration	12.5	7.3

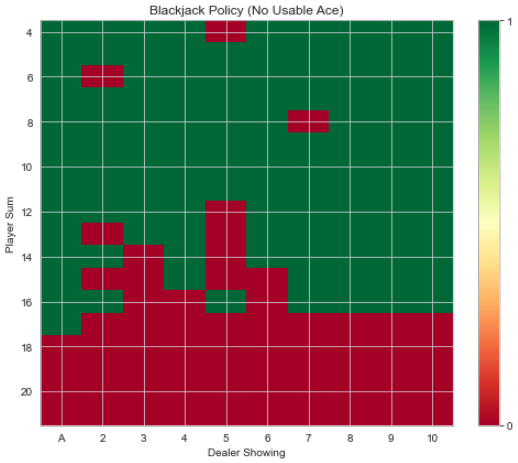

Fig. 4. Blackjack Heatmap: No Usable Ace

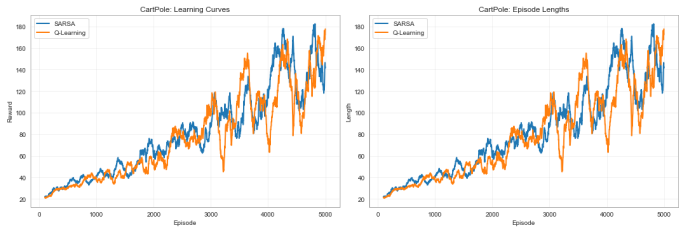

Fig. 5. CartPole LC and Episode Lengths

TABLE III
CARTPOLE PERFORMANCE WITH MEDIUM

Algorithm	Mean Reward	Std Dev	Training Time
Value Iteration	17.5	19.8	0.07s
Policy Iteration	19.5	21.9	0.10s
SARSA	70.8	34.6	23.29s
Q-Learning	321.2	72.0	24.77s

VI and PI achieved mean rewards of 17.5 and 19.5 respectively. This outlines the agents failure to learn the problem and effectively apply that to the issue of balancing the pole within the cart. High standard deviations further support this conclusion as the large variance suggest a complete failure to learn. This indicates that highly inconsistent performance, again a symptom of failure to learn and master the problem. Increasing the resolution of discretization had minimal impact suggesting this strategy failed to help agents learn the problem and effective decide around environment conditions. SARSA and Q-Learning performed much better. SARSA, despite the huge performance jump over VI/PI, netted lackluster performance with a mean reward of only 70.8. Q-Learning however, performed quite well with a mean score of 321.2, still with high variance. This would suggest that despite much better performance, this method did yield some domain knowledge but left a lot of room for improvement. It can also outline the more successful yet variable performance of exploration within policy making.

Considering learning curves of SARSA and Q-learning, shows good improvement over the 5,000 training episodes. Q-Learning was able to learn the policy much faster. Both

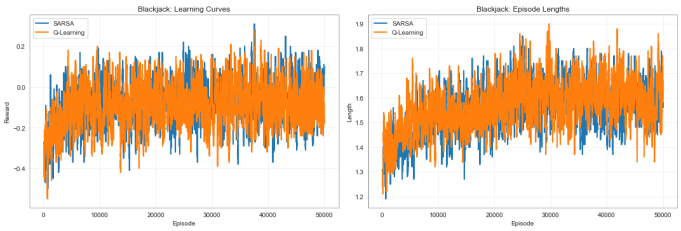

Fig. 6. Blackjack LC and Episode Lengths

models however show variability likely due to the ability to explore further, increasing variability, but ideally increasing agents ability to learn. Both curves suggest these methods could have benefited from additional training episodes or further improvements to parameters.

C. Analysis of Results

The results of these experiments seem to support the hypothesis, as in, agents were able to actually able to learn the CartPole environment. Despite high variability, and failure within VI/PI, SARSA and Q-Learning did show strong signs of learning the policy. Their performance indicated that further refinement is needed however, likely requiring additional training episodes or changes to parameters.

Despite some clear failures, the CartPole did produce results consistent with agents learning. The results of the blackjack environment outline the difficulty and stochastic nature of the problem, with agents unable to produce results consistent with learning the problem. In the case of blackjack, all agents performed similarly in regards to mean reward, but varied slightly in time to train. SARSA and Q-Learning did not meaningfully outperform VI/PI in that circumstance but did within the CartPole instance. Q-Learning proved to highly beneficial showing that off policy learning provided advantages to agents learning. Ultimately, discretization failed to help agents solve the CartPole environment, suggesting this methodology was a failure.

VI. CONCLUSION

This research provides an investigation into model based learners and model free based methods into two distinct environments. Blackjack and CartPole environment highlighted the differences between each method in a controlled manner, allowing for the analysis of algorithmic performance both across and within environments. Our hypothesis initial suggested that a deterministic environment would allow methods to properly exploit the predictable dynamics achieving larger performance scores than that of the stochastic environment, these results have yielded mixed results.

The results do appear to be consistent with that of the stochastic nature of the blackjack environment create too much uncertainty for the agents to properly master. However, a key detail we can directly observe is the method/model quality and its effect on the agents performance. VI/PI acheived near optimal performance within the blackjack environment (a score of 0 would be considered optimal) wheras in CartPole these methods failed entirely. Because of this, we

can reasonably say that the hypothesis does seem to be true. However, additional testing should be done in an attempt to remedy the weak performance displayed within CartPole. Despite issues in the implementation however, Q-Learning managed to approach the theoretical maximum of a reward of 500. This outlines that despite flaws in the implementation, off-policy decision making can learn to exploit the dynamics present within a deterministic environment. This features the strength of this methodology and potential use cases for its success.

These findings assert that model/method selection is crucial to a practitioners success within RL training and deployment. We can see clear positive results within blackjack for VI/PI, but terrible results within CartPole, similarly we can see mediocre results from SARSA and Q-Learning within blackjack, but decent results within CartPole. This outlines the importance of model/method selection at hand, and for the hyperparameter selection within them. Additionally methods such as discretization should be strongly evaluated and carefully designed so as not to hinder performance, which can be observed within these experiments. This showcases some unaccounted for nuance within RL methodologies. Performance may actually be much more associated with model quality rather than decision space or environment characteristics. Because of this, we neither reject nor accept the initial hypothesis, but rather affirm that additional work and research should be done to further explore, describe and remedy these nuances.

REFERENCES

[1] MDP: Sutton & Barto (2018), Reinforcement Learning: An Introduction (2nd ed.), Example 5.3 "Blackjack," online book. Gym Environment: Blackjack-v1.

[2] MDP: Barto, Sutton & Anderson (1983), "Neuronlike adaptive elements that can solve difficult learning control problems," IEEE Transactions on Systems, Man, and Cybernetics 13(5):834–846, doi:10.1109/TSMC.1983.6313077. Gym Environment: CartPole-v1.