Ayush Shrivastava (2016024) and Savit Gupta (2017098)

ARTIFICIAL INTELLIGENCE FINAL PROJECT SUBMISSION REPORT

Environment:

1. Implemented a blackjack environment with "n" decks that reset every "m" games.
2. The environment was implemented to replicate the functionality of an OpenAI gym environment allowing the user to reset the environment and get observable states at any point in time.

Q - Learning:

Why Q-Learning:

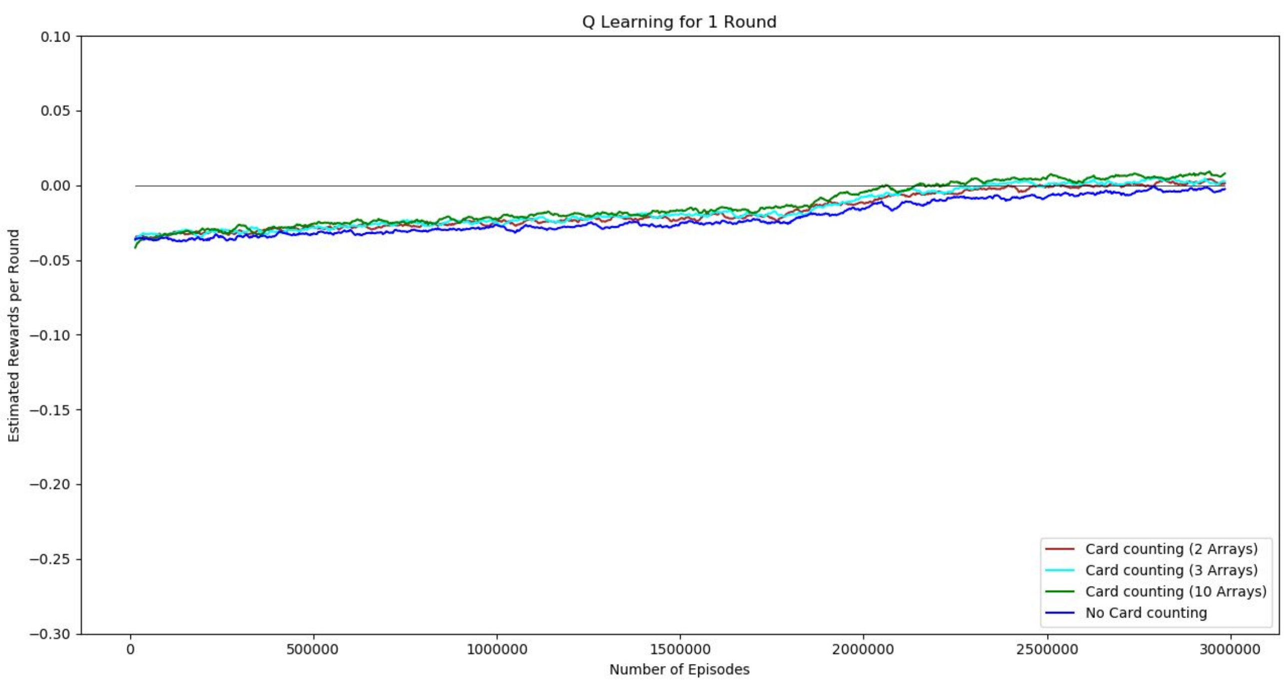
Blackjack is a game perfectly suited for the Q-learning algorithm due to its stochastic nature and a structure consisting of various game states, rewards and player actions.

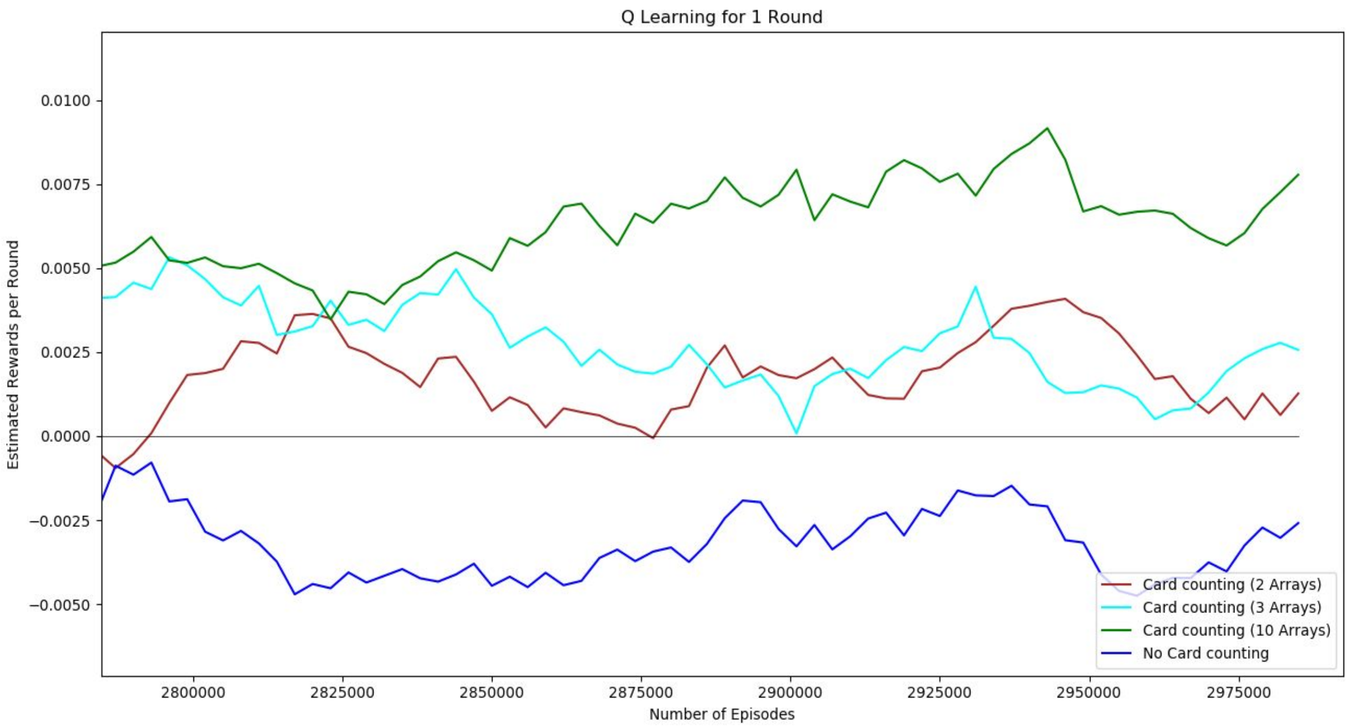
Game states:

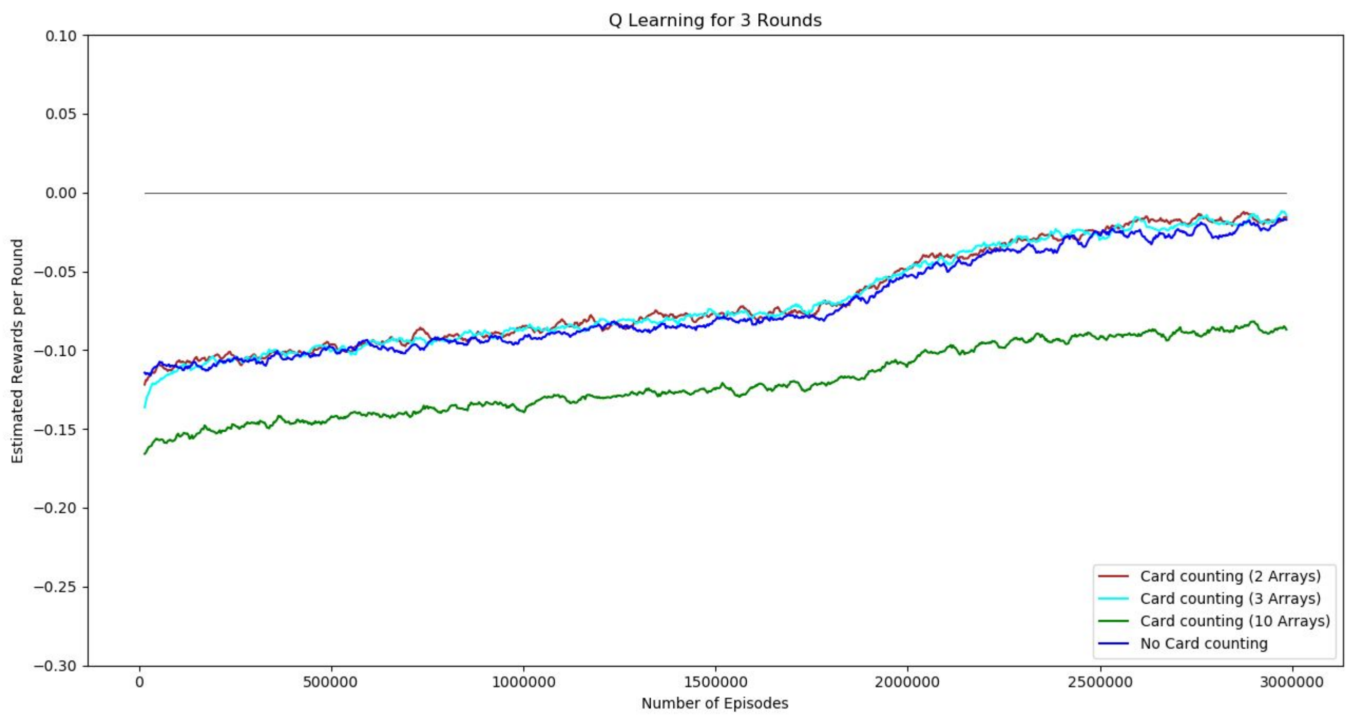
1. Game state without card counting: *(canUseAce, playerSum, dealerUpCard)*
2. Game state with card counting: *(canUseAce, playerSum, dealerUpCard, numRounds, countOfCards)*

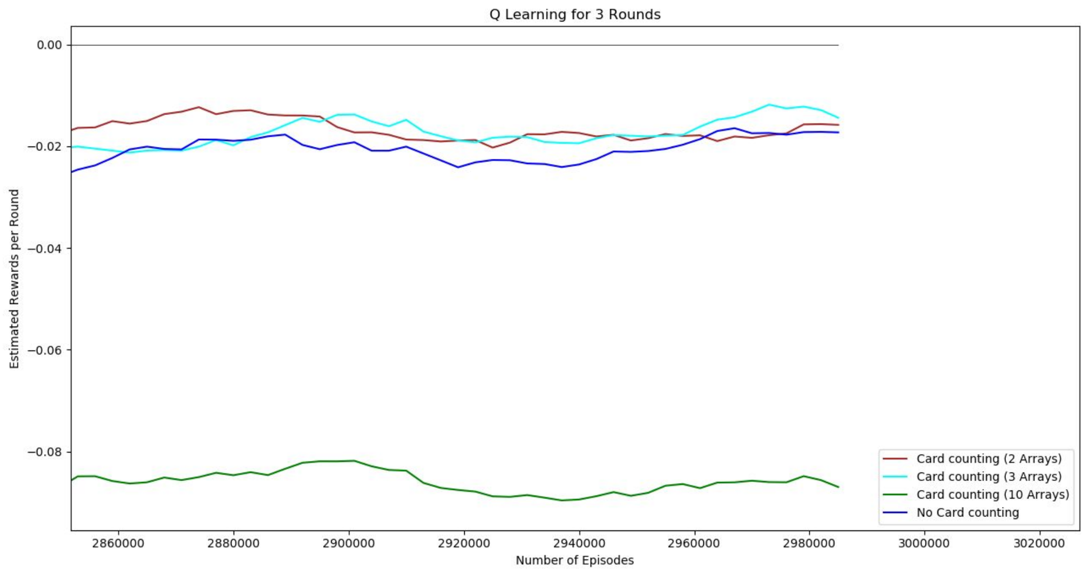
Observations:

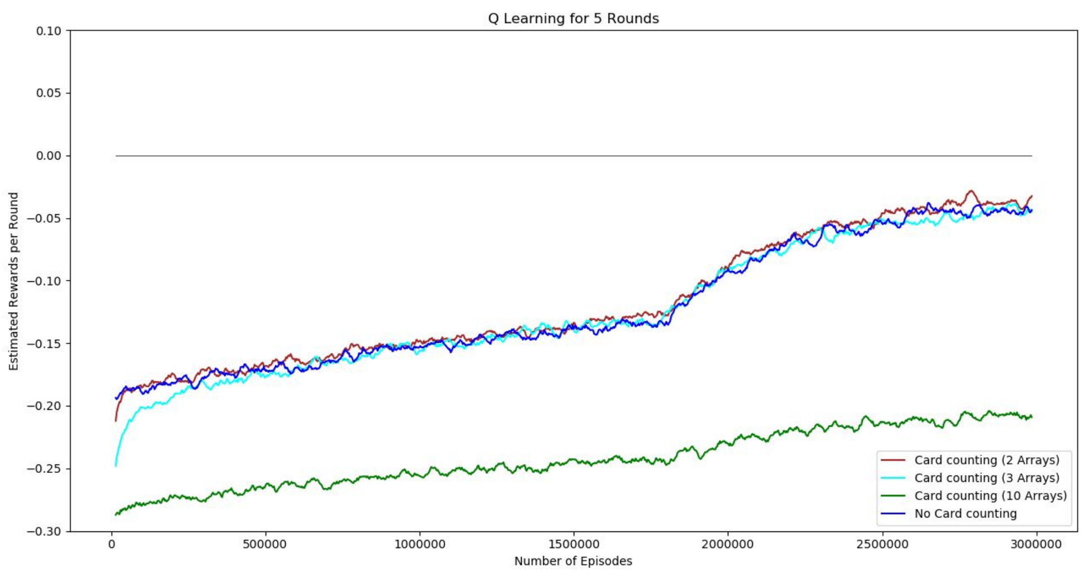
1. As *numRounds* increases the number of possible game states increases in a linear proportion.
2. As the number of arrays in which the *countOfCards* is split into increases, there is an exponential increase in the number of game states.
3. The knowledge space for the model on including these modifications to *numRounds* and *countOfCards* becomes very large, making it infeasible for optimal results to be calculated, thus leading to a tradeoff between feasibility and optimality of the algorithm.

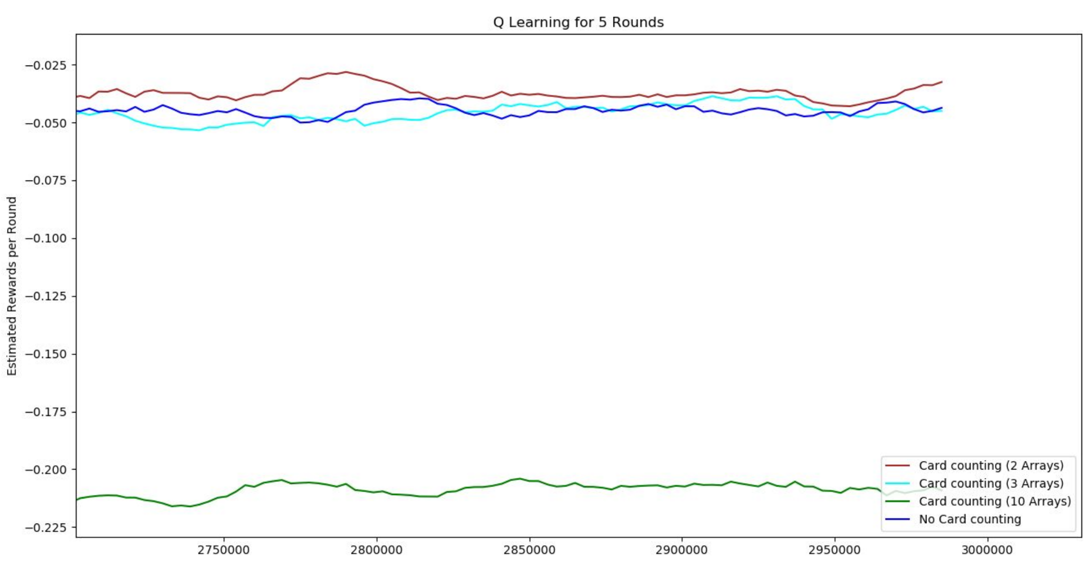



Q Learning for 1 Round
0.0100
0.0075
0.0050
0.0025
0.0000
−0.0025
−0.0050
Estimated Rewards per Round
2800000
2825000
2850000
2875000
2900000
2925000
2950000
2975000
Number of Episodes
Card counting (2 Arrays)
Card counting (3 Arrays)
Card counting (10 Arrays)
No Card counting


Q Learning for 3 Rounds
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
Estimated Rewards per Round
0
500000
1000000
1500000
2000000
2500000
3000000
Number of Episodes
Card counting (2 Arrays)
Card counting (3 Arrays)
Card counting (10 Arrays)
No Card counting


Q Learning for 3 Rounds
Estimated Rewards per Round
Number of Episodes
0.00
−0.02
−0.04
−0.06
−0.08
2860000
2880000
2900000
2920000
2940000
2960000
2980000
3000000
3020000
Card counting (2 Arrays)
Card counting (3 Arrays)
Card counting (10 Arrays)
No Card counting


Q Learning for 5 Rounds
Estimated Rewards per Round
Number of Episodes
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
−0.30
0
500000
1000000
1500000
2000000
2500000
3000000
Card counting (2 Arrays)
Card counting (3 Arrays)
Card counting (10 Arrays)
No Card counting


Q Learning for 5 Rounds
Estimated Rewards per Round
Number of Episodes
−0.025
−0.050
−0.075
−0.100
−0.125
−0.150
−0.175
−0.200
−0.225
2750000
2800000
2850000
2900000
2950000
3000000
Card counting (2 Arrays)
Card counting (3 Arrays)
Card counting (10 Arrays)
No Card counting

Deep Q - Learning:

Why Deep Q-Learning:

As observed in Q-Learning, when the number of states increases it becomes infeasible to get optimal values. Deep Q-Learning doesn't learn the exact values for every state, but rather approximates from the values that it learns.

Game states:

The game is only played with card counting and the game state is:

(canUseAce, playerSum, dealerUpCard, numRounds, countOfCards)

Observations:

1. Using DDQN rather than a DQN provides greater efficiency as it overcomes the "chasing own tail" problem due to continuously changing weights in the network.
2. As the number of episodes to train is increased, there is an exponential increase in the time taken by the deep Q network to train. Despite this, DDQN performs substantially better than standard Q-Learning, giving optimal results in much lesser episodes.