Reinforcement Learning

By Gabriel Hili (13502H), Joseph Grech (11602H), Charmaine Micallef (26102H)

Plagiarism Declaration Form

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as "the unacknowledged use, as one's own, of work of another person, whether or not such work has been published, and as may be further elaborated in Faculty or University guidelines" (University Assessment Regulations, 2009, Regulation 39 (b)(i), University of Malta).

We, the undersigned, declare that the assignment submitted is our work, except where acknowledged and referenced.

We understand that the penalties for committing a breach of the regulations include loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected, and will be given zero marks.

(N. B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).

Gabriel Hili	Gabriel Hili
Student Name	Signature
Joseph Grech	
Student Name	Signature
Charmaine Micallef	Charmaine Micallef
Student Name	Signature

ARI2204	Reinforcement Learning - Assignment
Course Code	Title of work submitted

22/04/2022
Date

Overview and Implementation

Blackjack Environment

The class *Game* encodes the Blackjack environment. The constructor of the class shuffles the deck consisting of 52 card points *[1,2,3,4,5,6,7,8,9,10,10,10,10]*4,* then gives 2 cards to the player and one to the dealer. The state of the game is represented by 3 variables in the class *State*. State has variables for the current player hand, the dealer's card and the boolean flag for the ace card.

As an episode is played, the game automatically hits if the player hand is less than 12, and stands if it's 21. Otherwise, the action to take is the one with the best Q-value given the current state, unless an explorative approach is used. Then the Q-values of the state-actions are updated based on the policy given. Montecarlo updates the values at the end of the episode while SARSA and Q-Learning update them on-line.

Reinforcement Learning Framework

Overview

Given some value for ε, a policy chooses whether to take an explorative or exploitative approach. An explorative approach means taking an action for the sake of broadening the knowledge about the game, regardless of maximising wins. An exploitative approach means taking an action that has empirically shown to produce the best results out of all other actions. Every policy (Montecarlo, SARSA, Q-Learning) chooses an explorative approach with probability ε; otherwise it takes on an exploitation strategy. The value for ε asymptotically approaches 0 as the number of episodes *k* increases. Hence, all policies employ an explorative strategy in the beginning but phase out into an exploitative one.

Implementation

Every state-action seen during the 500,000 episodes is stored in an array. The array can never be larger than 360 elements. This is because there are 180 different states and 2 possible actions from every state. Given an encountered state, the program attempts to find an instance of that state in the array in order to update its count. If it is not found then it is added.

```python
def increment_count(self, s:State, hit:bool) -> None:

    index = self.seen(s)

    if index != -1:
        self.model[index].increment_count(hit)
    else:
        index = len(self.episode)
        self.model.append(StateMetric(s,0,0,0,0).increment_count(hit))

    #Keep reference to position in model for O(1) look-up time.
    self.episode.append((s,hit,index))
```

The sequence of state-actions in an episode is stored, alongside the index of the state-action in the array. This is so when montecarlo policy updates all state-action of an episode, the state-actions are found in the model-array in O(1) time.

```python
def update_episode(self, win:bool):

    for (_,a,index) in self.episode:
        self.model[index].update_value(a,win)

    #Reset episode
    self.episode.clear()
```

Montecarlo will remember every state-action traversed during an episode and update the Q-value of each one at the end [1.] If the outcome is a win for the player then all Q-values are incremented by 1. If it's a loss then they are decremented by 1 and if it's a draw nothing is changed.

SARSA updates the Q-values on-line [2] and doesn't use the above function *update_episode*.

```
#Update previous state-action value using the current state-action and outcome.
if model.previous_state_action is not None:
    if lost: R = -1
    elif outcome == 'WIN': R = +1
    else: R = 0

    #Previous state-action
    previous_state = model.previous_state_action[0]
    previous_hit = model.previous_state_action[0]

    #We know that the previous state_action is seen.
    previous_index = model.seen(previous_state)
    previous_q_sa = model.model[previous_index].hit_value if previous_hit else model.model[previous_index].stand_value

    previous_hit_count   = model.model[previous_index].hit_count
    previous_stand_count = model.model[previous_index].stand_count

    current_index = model.seen(current_state)
    current_q_sa = model.model[current_index].hit_value if hit else model.model[current_index].stand_value

    if previous_hit: model.model[previous_index].hit_value   += (R + current_q_sa - previous_q_sa)/(1+previous_hit_count)
    else:            model.model[previous_index].stand_value += (R + current_q_sa - previous_q_sa)/(1+previous_stand_count)
```

The above function updates the current state-action value (S-A) using the reward received from that state-action (R) and the value of the next state-action (S-A). The hyperparameter α was set to be $\frac{1}{1+N(s,a)}$ where $N(s,a)$ is the number of times the current state-action was previously chosen by the SARSA policy. Below is the code snippet that updates the state-action value.

```
if previous_hit: model.model[previous_index].hit_value   += (R + current_q_sa - previous_q_sa)/(1+previous_hit_count)
else:            model.model[previous_index].stand_value += (R + current_q_sa - previous_q_sa)/(1+previous_stand_count)
```

Q-Learning is exactly the same as SARSA, in that it updates the current state-action value using the reward received from that state-action and the value of the next state-action. However, the value of the next state-action is the largest value of all possible actions (hit,stand) that can be taken from it, instead of the value of the actual action of the next state [3]. The code snippet below highlights the difference. We see that the previous state-action value is being updated using the best current state-action instead of the actual state-action chosen.

```
if previous_hit: model.model[previous_index].hit_value   += (R + best_current_q_sa - previous_q_sa)/(1+previous_hit_count)
else:            model.model[previous_index].stand_value += (R + best_current_q_sa - previous_q_sa)/(1+previous_stand_count)
```

Exploration vs Exploitation

Exploration is considered to be a long-term benefit concept as it allows the agent to improve its knowledge about all the actions that can eventually lead to long term benefit.

On the other hand, exploitation is when the agent's current estimated value is exploited and the greedy approach is chosen to get the most reward. However there is a chance that the

agent might get no reward at all since it's being greedy with the estimated value and not the real one.

Monte Carlo Algorithm

The agent opts for exploration rather than exploitation as it might not be beneficial to try multiple strategies to try and maximise the return. Making use of the random policy is the most beneficial as it will most likely explore all the possible actions from each state. However, after some more steps it made sense to opt for exploitation rather than exploration as the policy will gradually become more greedy. Ultimately, choosing exploration initially and then moving on to exploitation seemed to be the optimal strategy.

Sarsa

A random action with probability epsilon is taken as we do not wish to exploit the q-values greedily. So ideally the epsilon value will gradually decrease by the end of the learning process and in this case the total number of episodes.

Q-Learning

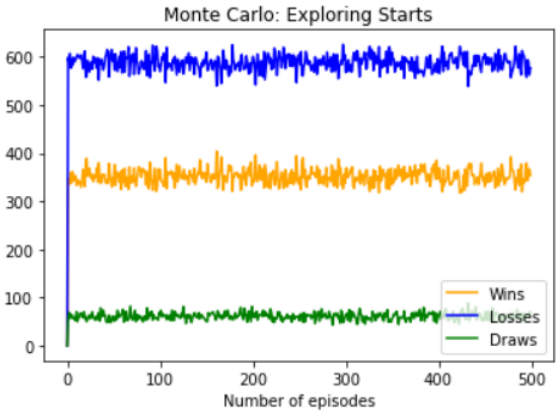
Here the agent is setting the value of epsilon to how often it is going to be explored vs exploited so as to become balanced. As q-learning is always looking to learn of a policy to maximise the reward, it looks to exploit. However, the algorithm does act randomly so it allows the agent to find out new states that it possibly would not have found before during the exploitation process.

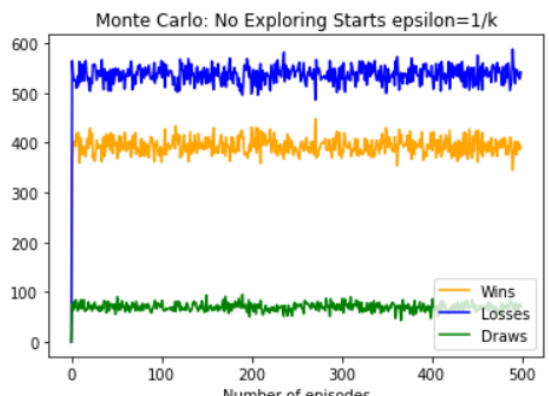
Evaluation

Monte Carlo

Configuration: $\varepsilon = \frac{1}{k}$

The configuration with *Exploring Starts* set to true resulted in the least number of wins.



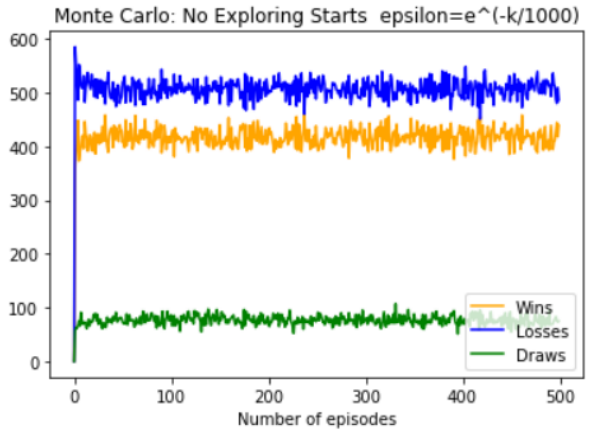


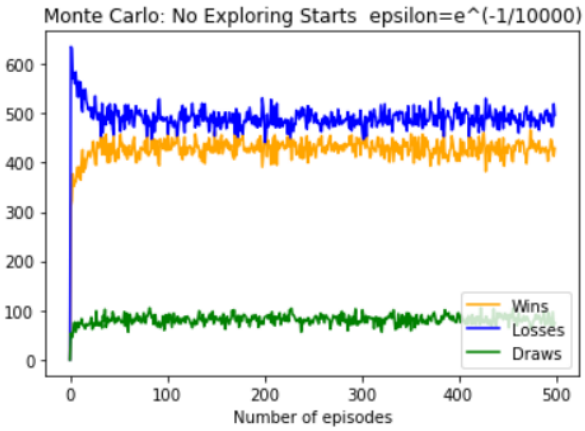
Configuration: $\varepsilon = \frac{1}{k}$

When *Exploring Starts* was set to False, there was an average increase of 50 wins per 1000 games.

Configuration: $\varepsilon = \frac{1}{e^{k/1000}}$

Initially, the average losses and wins seem to converge but then reach an equilibrium.

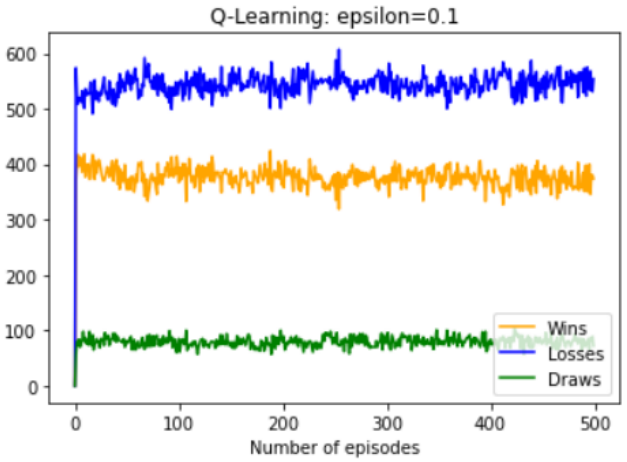




Configuration: $\varepsilon = \frac{1}{e^{k/10000}}$

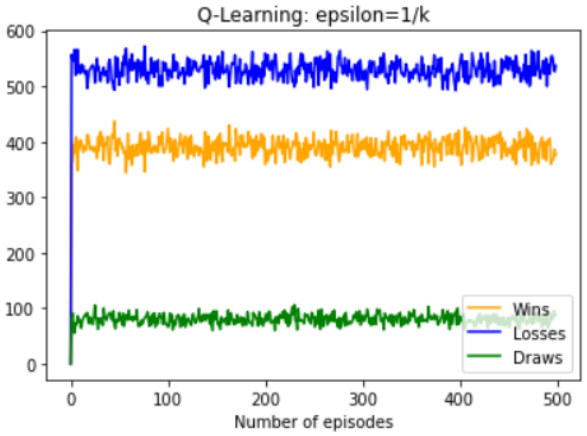
The convergence is more spread out unlike the previous configuration, which makes sense as the probability to employ an explorative approach reaches 0 more slowly then $\varepsilon = \frac{1}{e^{k/1000}}$.

Q-Learning



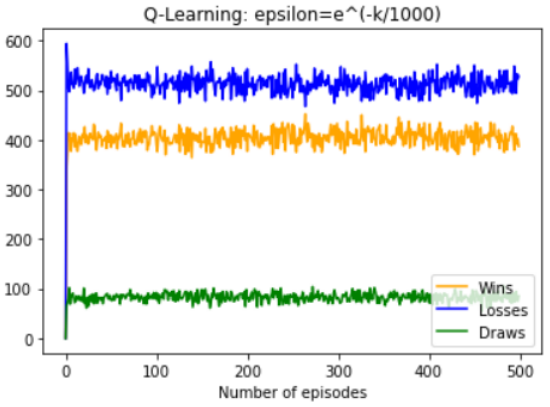
Configuration: $\varepsilon = \frac{1}{10}$

Initially the number of wins decreases and then converges to an average value. Similarly for the losses.



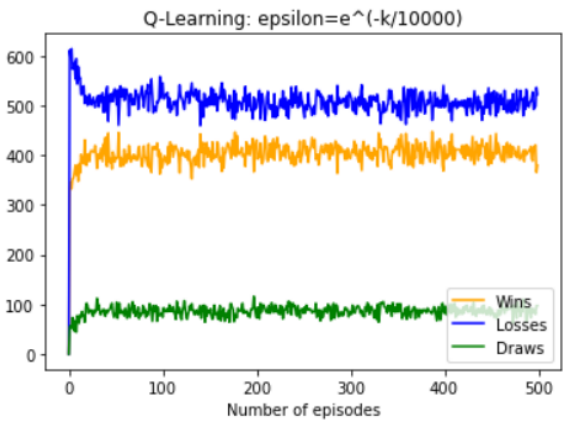
Configuration: $\varepsilon = \frac{1}{k}$

The number of losses and wins per 1000 episodes seem to be more uniform and converge immediately to around 550 and 400 respectively.



Configuration: $\varepsilon = \frac{1}{e^{k/1000}}$

In this configuration, the number of losses seem to be slightly lower than when $\varepsilon = \frac{1}{k}$.



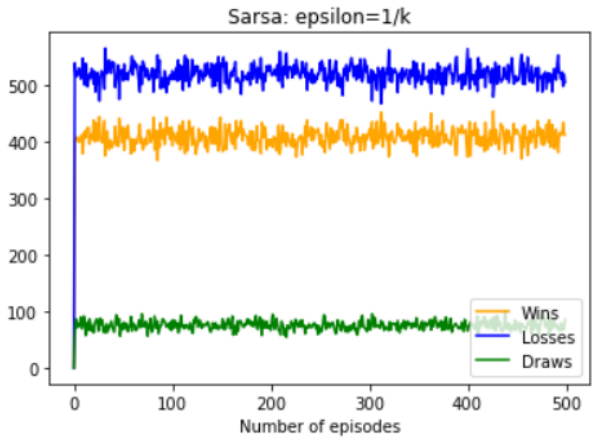
Configuration: $\varepsilon = \frac{1}{e^{k/10000}}$

The number of losses and wins converge towards each other before then flatlining. The number of draws also increases initially.

SARSA

Configuration: $\varepsilon = \frac{1}{10}$

The difference between the number of wins and losses is about 100 per 1000 episodes.

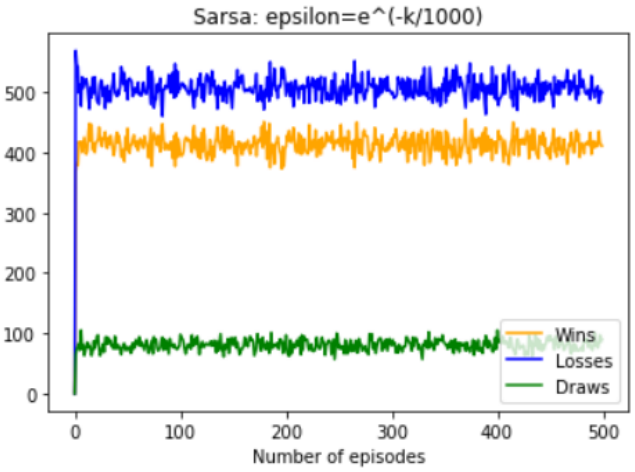


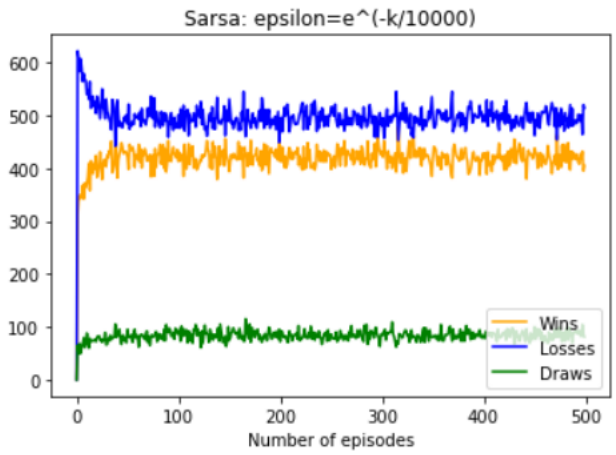
Configuration: $\varepsilon = \frac{1}{k}$

This configuration seems similar to when $\varepsilon = \frac{1}{10}$.

Configuration: $\varepsilon = \frac{1}{e^{k/1000}}$

The number of wins increased by about 25 every 1000 episodes. The gap between the number of wins and losses is less.



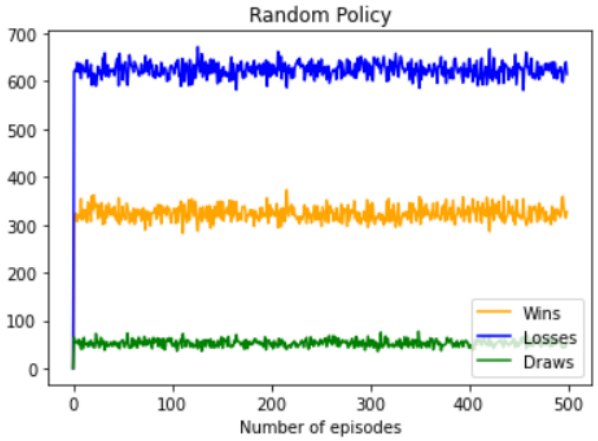


Configuration: $\varepsilon = \frac{1}{e^{k/10000}}$

Initially the number of wins and losses converge towards each other. This configuration has the lowest gap between the two.

Random Policy

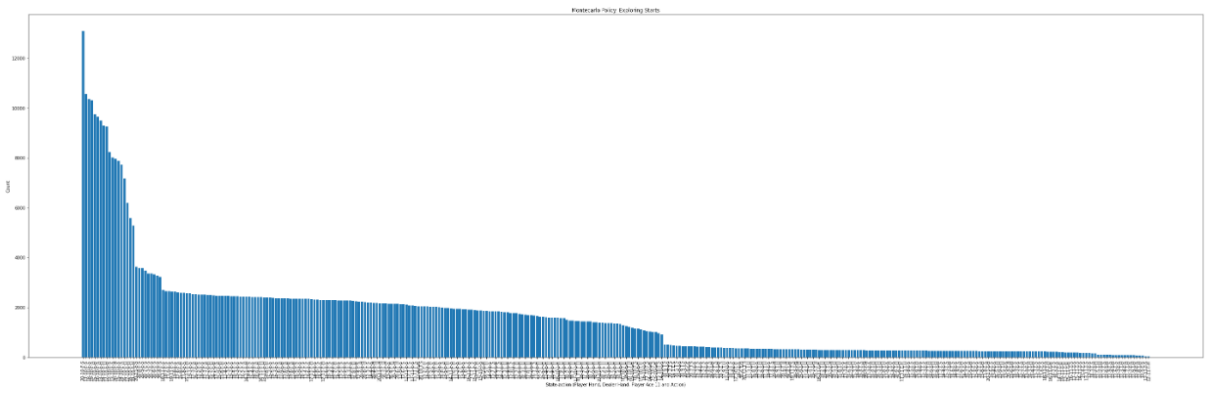
Employing a random policy gives the lowest number of wins, with a win rate of around 30%.



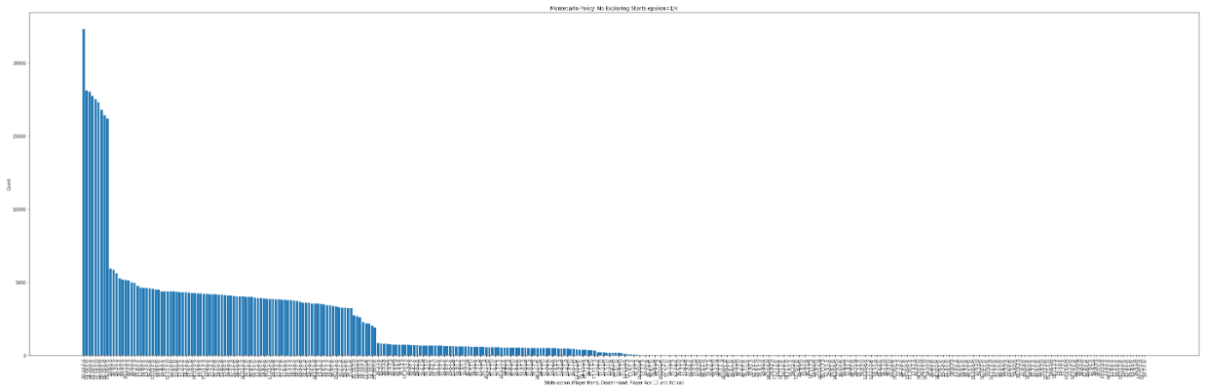
Unique State Action Counts

The graphs can be seen in higher quality in the Jupyter Notebook *Analysis.ipynb* The label represents a state-action. For example ***20-10-F-S*** means that the Player Sum was 20, the Dealer's Card was 10, the Player did not have an Ace acting as 11, and the action was to Stand.

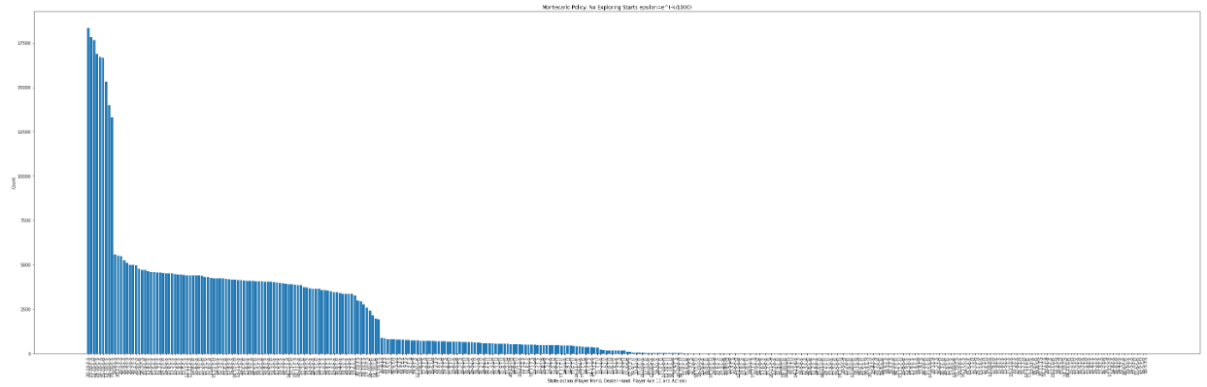
Monte Carlo



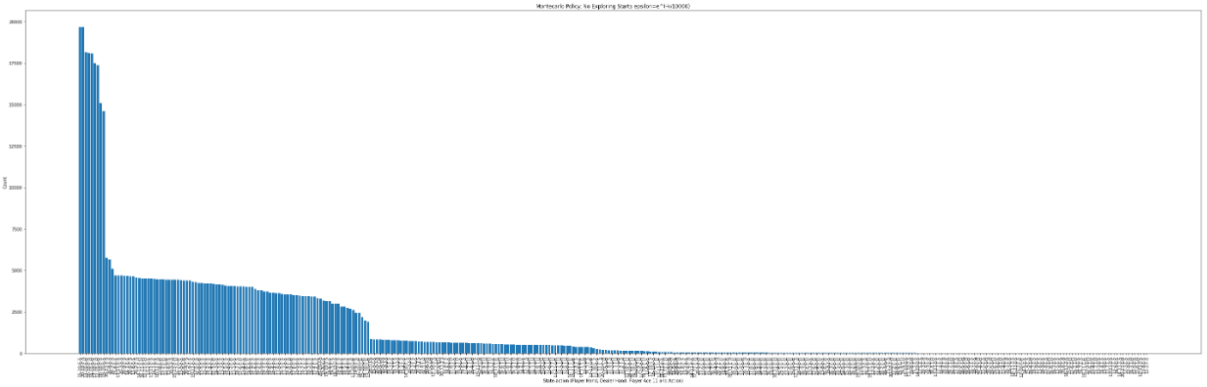
The highest unique state-action pair in this configuration with over 12000 counts is 20-10-F-S. It then drops suddenly and decreases in counts until 18-11-E-S when the decrease is more gradual. Another sudden drop at 20-6-T-S and then again decreases gradually.



In this configuration, the highest unique state-action pair with over 20000 counts is 20-10-F-S. There is a big drop to the second highest action-state, 13-10-F-H. It decreases gradually until the 10th highest action-state,20-8-F-S where there is a sudden drop and continues to decrease moderately. It drops for the last time with 20-2-T-S and decreases further.



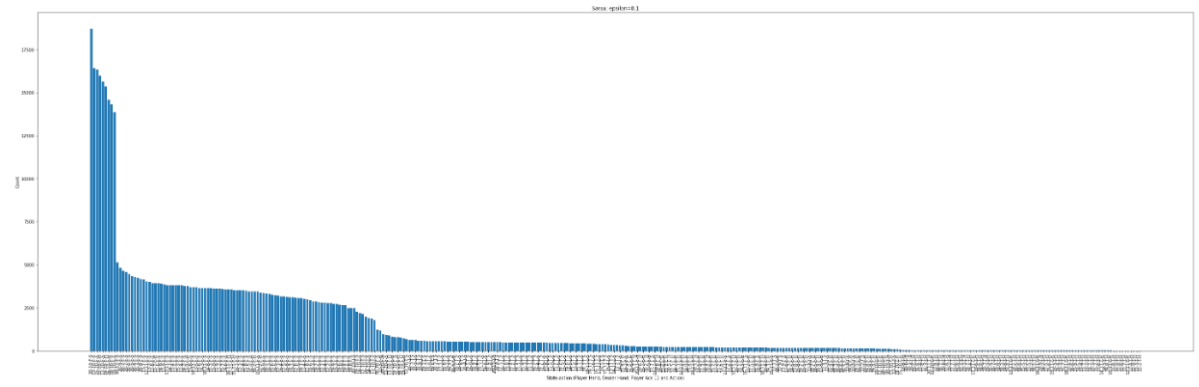
In this configuration, the highest unique state-action pair with over 17500 counts is 20-10-F-S. The pairs start to decrease in counts until a sudden drop to just over 5000 counts with 20-8-F-S being the 10th highest. The unique pairs start to decrease in counts until the last final drop with below 2500 counts, 20-4-T-S.



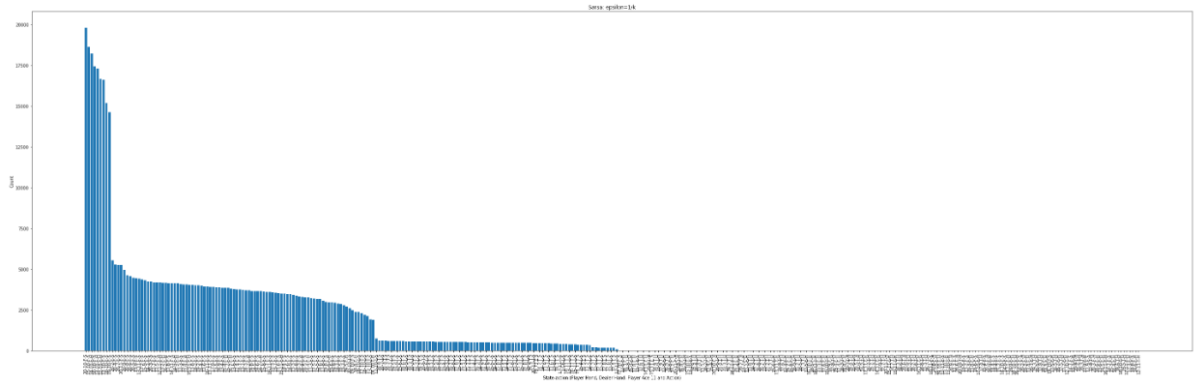
In this configuration, the highest unique action-state with the number of counts just below 20000 is 15-10-F-S. It starts to decrease in increments until a sudden drop to a little over

5000 counts with the state-action pair of 20-10-F-S being the 10th highest. It decreases moderately until a final drop with below 2500 counts with the state-action pair of 20-7-T-S.

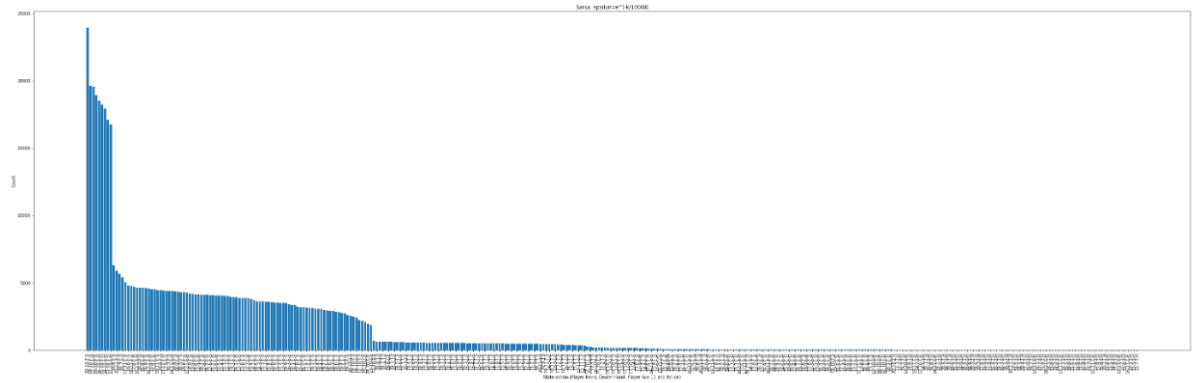
SARSA



In this configuration, the highest unique state-action pair with counts exceeding 17500 is 20-10-F-S. The counts drop to below 17500 for the 2nd highest pair and gradually decrease until a drop to around 5000 counts for the 10th highest pair of 20-11-F-S.It then decreases moderately until a small drop with the state-action pair of 16-10-F-S.

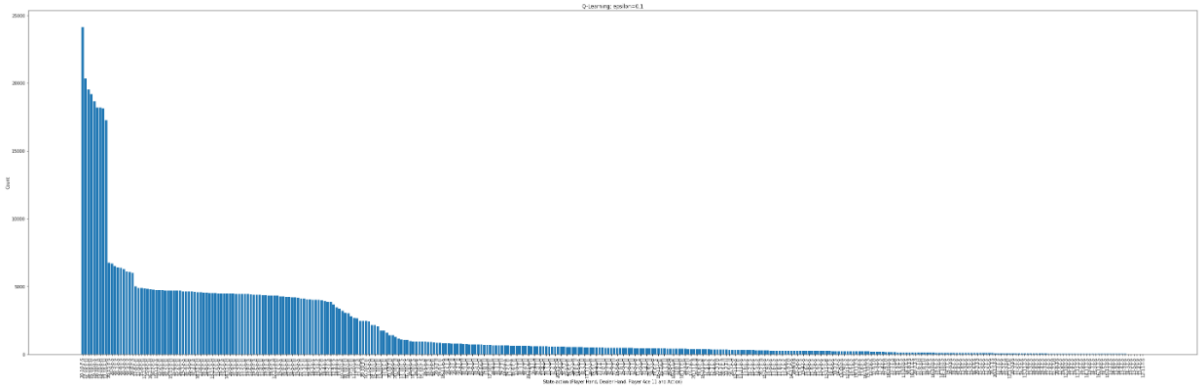


In this configuration, the highest unique state-action pair with counts nearly up to 20000 is 20-10-F-S. It then starts to drop to a little below 15000. Then it drops suddenly to just above 5000 counts with the state-action pair of 20-2-F-F-S being the 10th highest pair. It moderately decreases until a small drop to 1250 counts with the state-action pair 12-10-T-S.

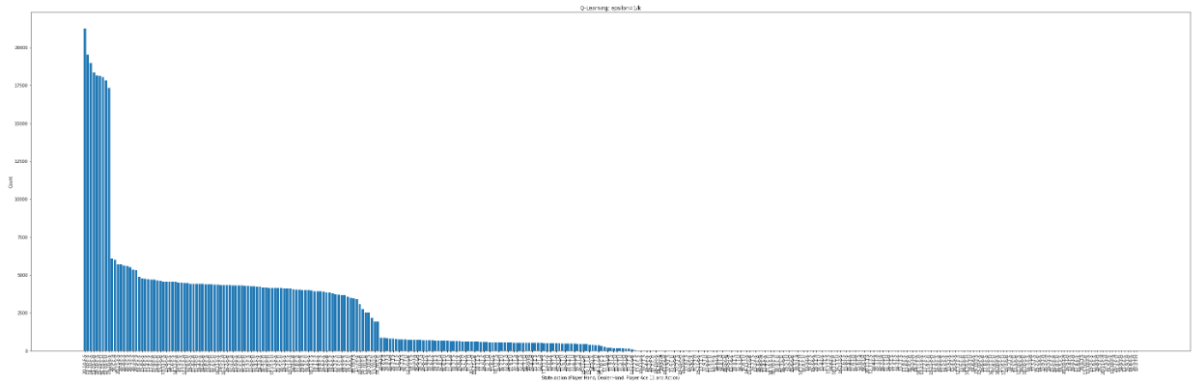


In this configuration, the highest state-action pair with counts almost at 25000 is 20-10-F-S. It then drops by around 5000 counts making the 2nd highest state-action pair 18-10-F-S. It continues to decrease in increments until another major drop to a little above 5000 counts with 20-8-F-S being the 10th highest. It continues to decrease slowly until the last small drop with counts below 2500 with the state-action pair of 12-10-T-S.

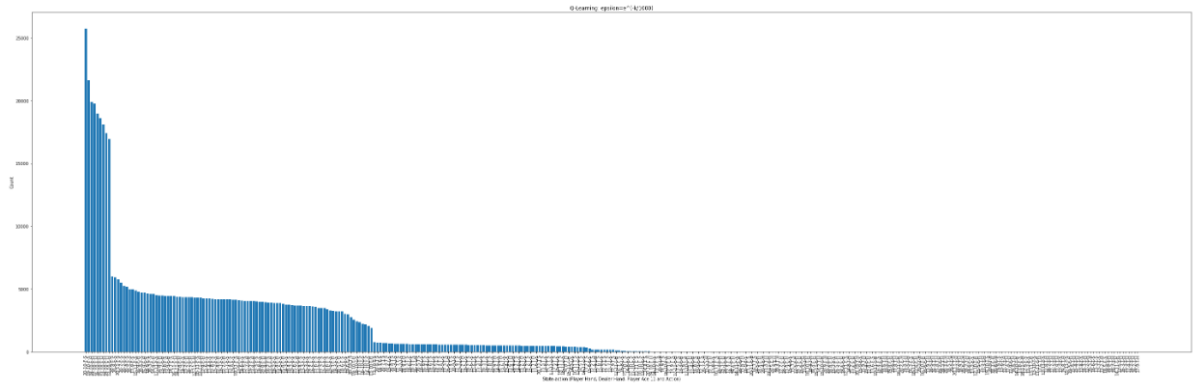
Q-Learning



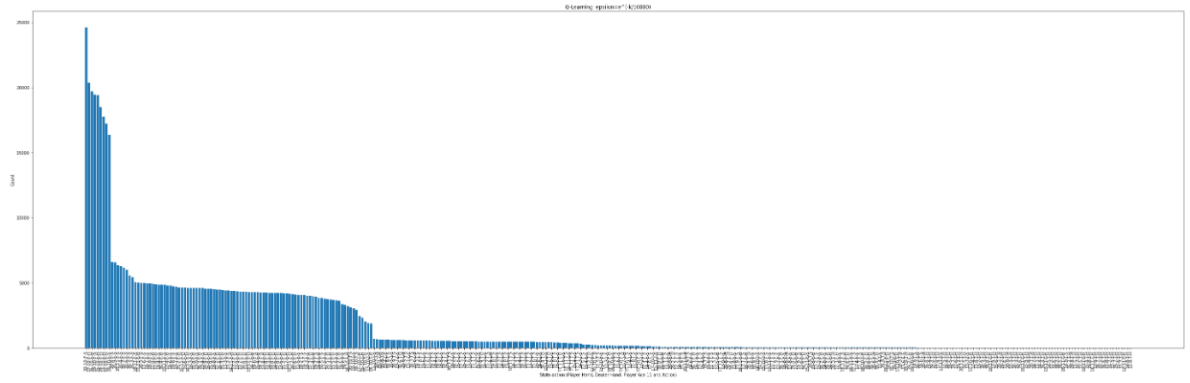
In this configuration, the highest unique action-state pair with just below 25000 counts is 20-10-F-S. The counts drop to around 20000 for the 2nd highest pair of 17-10-F-S and decrease slowly until a major drop. 20-11-F-S being the 10th highest pair with above 5000 counts. It continues to decrease slowly for the next 8 pairs and then there is a little drop to around 5000 counts with 20-6-F-S being the 19th highest pair. It decreases in counts very slowly and then gradually.



In this configuration, the highest action-state pair is 20-10-F-S with counts much above 20000. It drops to below 20000 counts for the 2nd highest pair and then gradually decreases until a major drop of just above 5000 counts with the 10th highest state-action pair of 20-9-F-S. Counts gradually decrease in counts until a small drop to below 2500 counts with the pair 20-4-T-S and then continues to gradually decrease.



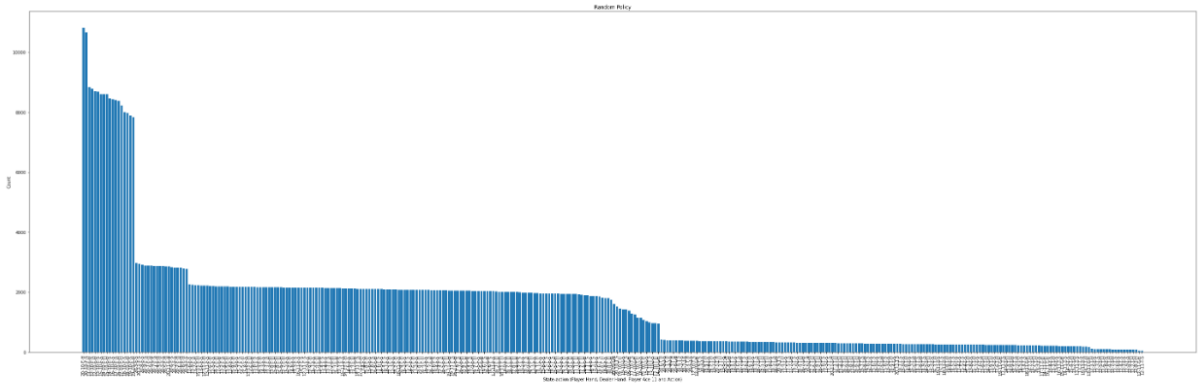
In this configuration, with over 25000 counts the highest action-state pair is 20-10-F-S. Counts then drop to just above 20000 for the 2nd highest pair of 19-10-F-S. The counts decrease until a major drop to just above 5000 counts for the 10th highest pair of 20-9-F-S. It decreases moderately until a small drop with the action-state pair of 12-10-T-S at below 2500 counts.



In this configuration, the highest action-state pair is 20-10-F-S with counts just below 25000. Counts decrease to a little above 20000 for the 2nd highest pair of 17-10-F-S. It continues by

decreasing gradually until a sudden decrease to a bit above 5000 counts with 20-8-F-S being the 10th highest pair. It moderately decreases in counts until a small drop of below 5000 counts with the state-action pair 12-10-T-S.

Random Policy



In this configuration, the Ace is played as 1. The highest state-action pair is 20-10-F-H with over 10000 counts. Then a small decrease at the 3rd highest pair with a little over 8000 counts, 13-10-F-H. There is a gradual decrease in counts until a sudden drop to around 3000 counts with 20-3-F-S being the 19th highest pair. There is a small drop later on with around 2000 counts with the pair being 12-6-F-S, 37th highest. After gradually decreasing, there is a small drop with less than 1000 counts, the pair being 20-6-T-H.

Total Unique State Action Pairs

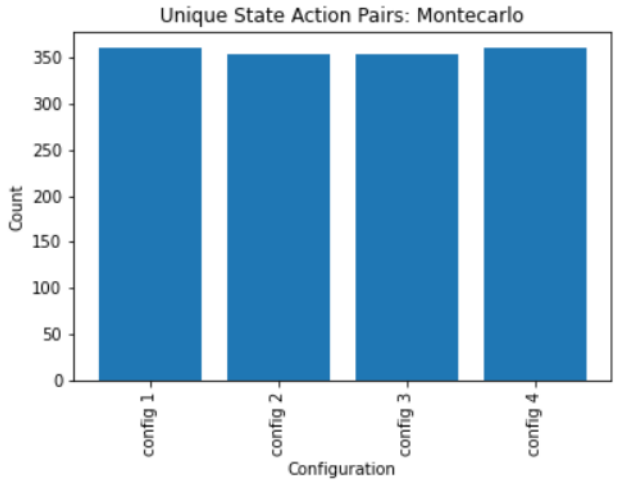
Monte Carlo

Config-1: *exploring starts* = *True* and $\varepsilon = \frac{1}{k}$

Config-2: *exploring starts* = *False* and $\varepsilon = \frac{1}{k}$

Config-3: $\varepsilon = \frac{1}{e^{k/1000}}$

Config-4: $\varepsilon = \frac{1}{e^{k/10000}}$



All configurations for Monte Carlo come close to the 360 different configurations limit. The lowest is *config-2* which is *exploring starts* = *false* and $\varepsilon = \frac{1}{k}$

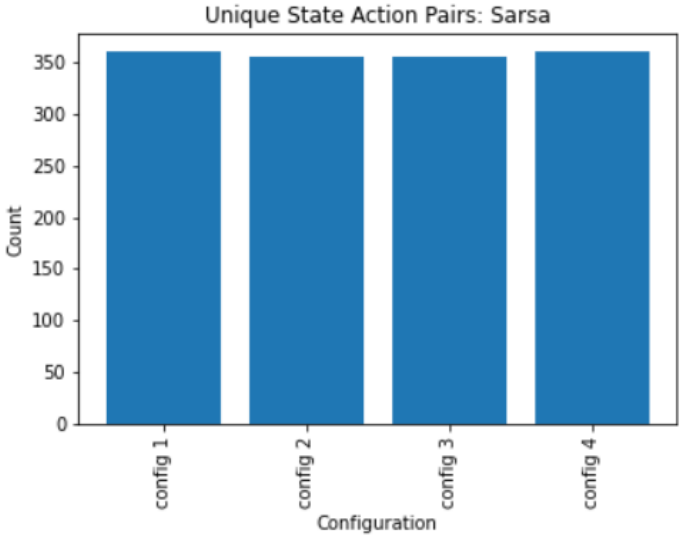
Sarsa

Config-1: $\varepsilon = \frac{1}{10}$

Config-2: $\varepsilon = \frac{1}{k}$

Config-3: $\varepsilon = \frac{1}{e^{k/1000}}$

Config-4: $\varepsilon = \frac{1}{e^{k/10000}}$



Configurations 1 and 4 and configurations 2 and 3 are similar. Configurations 2 and 3 have slightly less counts than configurations 1 and 4.

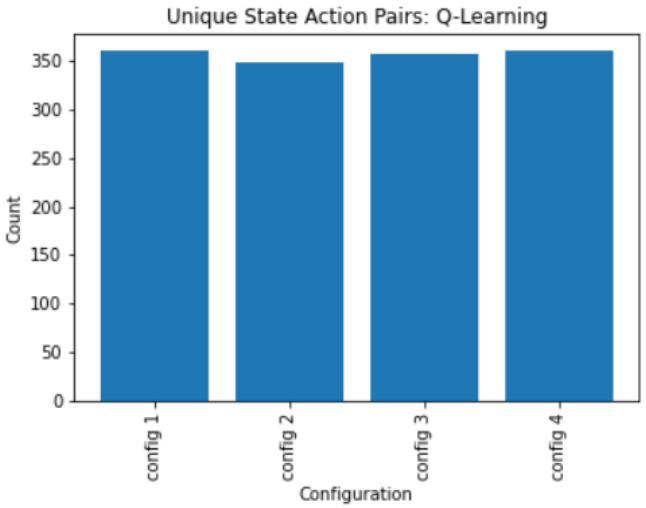
Q-Learning

Config-1: $\varepsilon = \frac{1}{10}$

Config-2: $\varepsilon = \frac{1}{k}$

Config-3: $\varepsilon = \frac{1}{e^{k/1000}}$

Config-4: $\varepsilon = \frac{1}{e^{k/10000}}$



Configurations 1 and 4 are similar while configuration 3 has slightly less counts. Configuration 2 is the one with the least counts.

Blackjack Strategy Table

Every configuration listed has 2 tables. One for when ace-11 is false and the other when it's true. Generally, the higher the player sum, the more likely it is for the best action to be a STAND, rather than a HIT.

Monte Carlo

Config-1: *exploring starts* = *True* and $\varepsilon = \frac{1}{k}$

Config-2: *exploring starts* = *False* and $\varepsilon = \frac{1}{k}$

Config-3: $\varepsilon = \frac{1}{e^{k/1000}}$

Config-4: $\varepsilon = \frac{1}{e^{k/10000}}$

Config-1

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
17	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	H	H	H	S	H
15	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
12	H	S	S	S	H	H	H	H	H	H

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	H	H	S	S	S	S	H	H	H	S
17	H	H	H	H	H	H	H	H	H	H
16	H	H	H	H	H	H	H	H	H	H
15	H	H	H	H	H	H	H	H	H	H
14	H	H	H	H	H	H	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H

Config-2

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	H	H	S	H	S	S	S	H	S	S
18	S	S	H	H	S	S	S	S	S	H
17	H	S	S	H	S	S	S	H	H	S
16	S	H	S	S	S	S	H	H	H	S
15	S	H	S	S	H	H	S	S	S	S
14	H	H	H	S	H	S	H	S	S	S
13	S	S	H	S	S	S	H	H	H	S
12	S	S	S	S	S	H	H	H	H	H

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	H	S	H	S	H	H
19	S	H	S	S	S	H	S	S	S	S
18	S	S	S	S	S	H	S	S	S	H
17	H	H	S	S	S	H	S	H	H	H
16	H	S	H	H	S	S	H	S	S	H
15	H	S	H	H	S	H	S	H	S	S
14	H	H	S	S	H	S	S	H	S	S
13	H	S	H	S	H	H	H	S	H	S
12	H	S	H	H	H	H	H	S	S	S

Config-3

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	H	S	S	S
17	S	S	S	S	S	S	H	S	S	S
16	S	S	S	H	S	S	S	S	S	H
15	S	H	S	H	S	S	S	H	H	H
14	H	S	H	S	H	S	H	S	S	H
13	S	S	S	H	H	H	H	H	S	S
12	H	S	H	S	S	H	S	H	H	S

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	H	S	S	S	S	S	H
19	S	H	S	H	S	S	S	H	S	S
18	H	H	S	H	H	S	S	H	S	S
17	H	S	H	H	S	S	H	H	S	H
16	S	H	H	S	S	H	H	S	H	H
15	S	H	H	S	H	H	H	H	H	H
14	S	H	H	H	S	H	H	H	H	S
13	S	S	H	S	H	H	H	H	S	S
12	S	H	H	S	H	H	H	S	H	S

Config-4

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
17	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	S	H	H	S	H
15	S	S	S	S	S	H	S	S	S	H
14	S	S	S	S	S	S	H	S	H	H
13	S	S	S	S	S	H	H	S	H	S
12	H	H	S	S	H	H	H	S	H	H

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	H	H	S	S	S	H	S	H	S	H
17	S	S	H	H	S	H	H	H	H	H
16	H	S	H	H	H	H	S	H	H	H
15	H	H	S	H	S	H	H	H	H	H
14	H	S	H	H	S	H	H	H	H	H
13	H	H	H	H	H	S	H	H	H	H
12	H	H	S	H	H	H	S	H	S	H

SARSA

Config-1: $\varepsilon = \frac{1}{10}$

Config-2: $\varepsilon = \frac{1}{k}$

Config-3: $\varepsilon = \frac{1}{e^{k/1000}}$

Config-4: $\varepsilon = \frac{1}{e^{k/10000}}$

Config-1

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
17	S	S	S	S	S	S	S	S	S	H
16	H	S	S	S	S	S	S	H	H	H
15	S	S	S	S	S	S	S	S	S	H
14	S	S	S	S	S	S	S	S	H	S
13	S	S	S	S	S	S	S	H	S	S
12	H	S	S	S	S	H	S	S	H	S

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
17	S	S	S	S	S	S	H	S	S	S
16	S	S	S	S	S	S	S	S	S	S
15	S	S	S	S	S	S	S	S	S	S
14	S	S	S	S	S	S	S	S	S	S
13	S	S	S	S	S	S	S	S	S	S
12	S	S	S	S	S	S	S	S	S	S

Config-2

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	H	S	S	S	S	S	H
18	S	S	H	S	S	S	H	S	S	S
17	S	S	S	S	S	S	S	S	S	S
16	H	S	S	S	H	H	S	S	S	S
15	H	S	H	S	S	H	S	S	S	H
14	S	S	H	S	S	H	S	H	H	S
13	H	S	S	S	S	S	S	S	H	S
12	H	S	S	S	S	S	S	H	H	H

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
17	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	S	S	S	S	S
15	S	S	S	S	S	S	S	S	S	S
14	S	S	S	S	S	S	S	S	H	S
13	S	S	S	S	S	S	S	S	S	S
12	S	S	S	S	S	S	S	S	S	S

Config-3

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
17	S	S	S	H	S	S	H	S	H	S
16	S	S	H	S	S	S	S	S	H	S
15	S	S	S	S	S	H	H	S	H	S
14	S	H	S	H	S	S	S	S	H	H
13	S	H	H	S	S	S	H	S	H	H
12	S	S	S	H	H	H	H	H	H	H

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
17	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	S	S	S	H	S
15	S	S	S	S	S	S	S	S	S	S
14	S	S	S	S	S	S	S	S	S	S
13	S	S	S	S	S	S	S	S	S	S
12	S	S	S	S	S	S	S	S	S	S

Config-4

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
17	S	S	S	S	S	S	H	S	H	S
16	H	S	S	S	S	H	H	S	H	H
15	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
12	S	S	S	S	S	S	H	H	H	H

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
17	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	S	S	S	S	S
15	S	S	S	S	S	S	S	S	S	S
14	S	S	S	S	S	S	S	S	H	S
13	S	S	S	S	S	S	S	S	S	S
12	S	S	S	S	S	S	S	S	S	S

Q-Learning

Config-1

Player Sum	2	3	4	5	6	7	8	9	10	11
20	H	H	H	H	H	H	H	H	S	H
19	H	H	H	H	H	H	H	H	H	H
18	H	H	H	H	H	H	H	H	H	H
17	H	H	H	H	H	H	H	H	H	H
16	H	H	H	H	H	H	H	H	H	H
15	H	H	H	H	H	H	H	H	H	H
14	H	H	H	H	H	H	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	H	S	S	S	S	S	S	S	S	S
18	H	S	S	H	S	S	S	S	S	H
17	H	H	S	H	S	H	H	H	H	H
16	H	H	H	H	H	H	H	H	H	H
15	H	H	H	H	H	H	H	H	H	H
14	H	H	H	H	H	H	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H

Config-2

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	H	S	S	S	S	S	S
19	S	H	S	S	S	S	S	S	S	S
18	S	S	S	H	S	H	S	S	S	S
17	H	S	H	S	S	S	H	H	S	H
16	H	H	H	S	S	S	S	H	H	S
15	H	H	H	H	S	S	S	H	H	H
14	S	H	S	H	H	S	H	H	S	S
13	S	S	S	H	S	H	H	S	H	S
12	H	S	H	H	H	H	H	H	H	H

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	H	S	S	S	S	H
19	S	S	H	S	S	H	S	S	H	H
18	S	S	S	S	H	S	H	S	H	S
17	H	S	H	H	S	S	S	H	H	S
16	H	S	H	S	H	H	S	S	S	H
15	S	H	H	S	H	S	H	S	H	H
14	H	S	H	H	H	H	H	S	H	H
13	H	H	H	S	H	H	H	S	S	H
12	S	H	H	H	H	H	H	S	H	H

Config-3

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	S	S	S	H	S	S	S	S	H	S
17	H	S	S	S	S	S	H	S	H	S
16	S	H	H	S	S	H	H	H	H	H
15	S	S	H	S	S	H	H	H	H	S
14	S	S	S	S	S	S	H	H	H	H
13	H	S	S	H	H	H	H	H	H	H
12	H	S	H	S	S	H	S	H	H	H

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	H	H	S
18	S	S	S	S	S	S	S	S	S	S
17	H	S	S	S	S	S	S	S	S	S
16	S	S	S	S	H	H	H	S	S	S
15	S	S	S	S	S	H	S	S	S	H
14	S	S	H	S	S	S	S	H	S	S
13	S	H	S	S	H	H	S	S	H	S
12	S	S	H	S	S	S	S	S	S	H

Config-4

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	S	S	S	S	S	S
19	S	S	S	S	S	S	S	S	S	S
18	S	S	S	S	S	S	S	S	S	S
17	S	S	H	S	S	H	H	H	H	H
16	H	S	H	S	H	H	H	H	H	H
15	H	H	H	S	H	H	H	H	H	H
14	S	H	H	H	H	H	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
12	H	H	H	S	H	H	H	H	H	H

Player Sum	2	3	4	5	6	7	8	9	10	11
20	S	S	S	S	H	S	H	S	S	S
19	S	S	S	S	S	S	S	H	S	S
18	S	S	S	S	H	S	S	S	S	S
17	S	S	H	S	S	S	S	S	H	H
16	S	S	S	S	S	H	S	H	H	S
15	S	S	S	S	S	S	S	S	H	S
14	S	S	S	S	S	S	S	H	H	S
13	H	S	S	S	S	S	S	S	S	S
12	S	S	S	S	S	H	S	S	S	S

Dealer Advantage

Monte Carlo

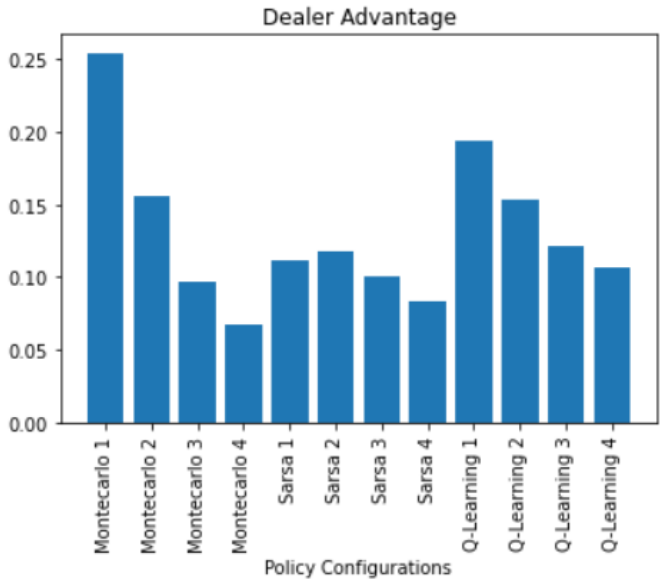
	Exploring Starts ε = 1/k	No Exploring Starts ε = 1/k	ε = e^(-k/1000)	ε = e^(-k/1000)
Mean Wins	349.89	392.77	417.51	428.47
Mean Losses	588.78	538.05	507.04	489.66
Mean Draws	61.33	69.18	75.45	81.87
D.A.	0.2545	0.1561	0.0968	0.0666

SARSA

	ε = 0.1	ε = 1/k	ε = e^(-k/1000)	ε = e^(-k/1000)
Mean Wins	411.29	407.87	413.07	419.82
Mean Losses	514.74	517.03	505.35	495.7
Mean Draws	73.97	75.1	81.58	84.48
D.A.	0.1117	0.1180	0.1005	0.0829

Q-Learning

	ε = 0.1	ε = 1/k	ε = e^(-k/1000)	ε = e^(-k/1000)
Mean Wins	371.05	390.08	403.54	407.52
Mean Losses	548.93	531.03	514.55	504.94
Mean Draws	80.02	78.89	81.91	87.54
D.A.	0.1934	0.1530	0.1209	0.1068



Lowest dealer advantage is obtained using the 4th configuration of the Montecarlo Policy (No exploring starts, epsilon = e^(-k/10000).

Distribution of work

Gabriel Hili:

- Blackjack environment.
- Reinforcement learning algorithms
- Report

Joseph Grech:

- Reinforcement learning algorithms
- Analysis of data
- Report

Charmaine Micallef:

- Reinforcement learning algorithms
- Evaluation of data
- Report

Citations

[1] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo method*. Hoboken, N.J.: Wiley. Copyright, 2017.

[2] G. A. Rummery and M. Niranjan, *On-line q-learning using connectionist systems*. Cambridge Univ. Of Cambridge, Department Of Engineering, 1994.

[3] C. J. C. H. Watkins, “Learning from delayed rewards.,” *ethos.bl.uk*, 1989. https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.330022 (accessed May 02, 2022).