RESIT Final Homework Assignment

Julia Sudnik, 2715115

1 Multi-Armed Bandits

1.1 Thompson Sampling for Single bandit

1.1.1 Implementation

Firstly, I have implemented the beta density function myself basing on the equation provided in the assignment. I have used the idea for distinguishing the bell shape and straight line types of beta-density graphs from the following article on towardsdatascience. The code used for plotting the distributions was also inspired by the one published in the mentioned article. Nonetheless, it was further extended to present different examples. In total, 4 plots were generated, each with 3 different values of alpha and beta to demonstrate differences in several aspects.

For the Thompson update, I found it difficult to understand what was meant in the assignment by stating that the p value is unknown. In the end, I understand that the p value is unknown from the Thompson update perspective, nonetheless it was necessary to set it to a certain value in order to use the Thompson update function. For the means of experimenting with the variance and mean I have set the probability of success to 0.5. Within the experiment, the Thompson update was iterated 140 times, due to received error "int too large to convert to float", when trying to increase the value. For this reason in the next parts of the assignment I have used a beta function from the scipy.stats module.

1.1.2 Findings

Several finding on the properties of the Beta-density distribution will be presented below. Firstly, I will address the differences pointed out in the assignment, that is, how the distribution vary depending on the relation between α and β.

- As shown in Figure 1 (look Beta(5,5), if $\alpha = \beta$, then the distribution is symmetric with a peak in $x = 0.5$. Moreover if $\alpha = \beta = 1$, a uniform distribution $y = 1$ is formed (see Figure 2 Beta(1,1)).
- Further, if $\alpha > \beta$, the distribution is right leaning (See Figure 1, 2: blue plots Beta(8,2), Beta(2,1)).
- Similarly if $\alpha > \beta$, the distribution is left leaning (See Figure 1, 2: red plots Beta(2,8), Beta(1,2))

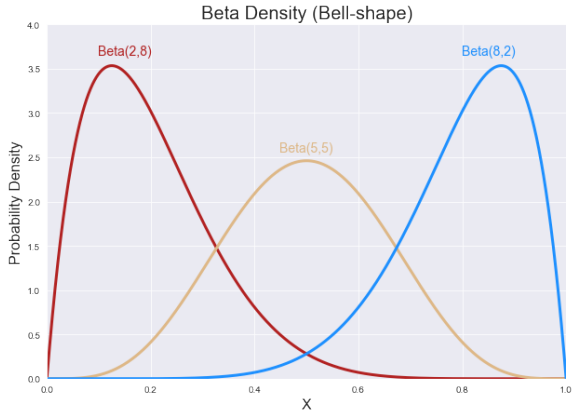

Figure 1: Beta density distribution

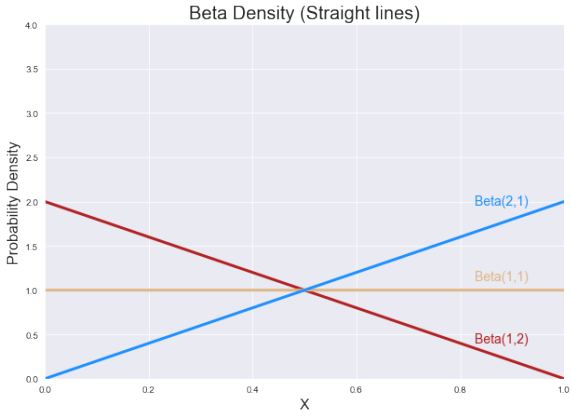

Figure 2: Beta density distribution

Moreover I have decided to explore more characteristics of the distribution and the following has been found.

- In general, the higher the α and the β, the more peaked is the distribution (See Figure 1.
- The larger the difference between α and β, the more squeezed the distribution is (See Figure 4.
- Although mostly curved, the distribution forms straight lines if $\alpha = 1$ or $\beta = 1$, or both.

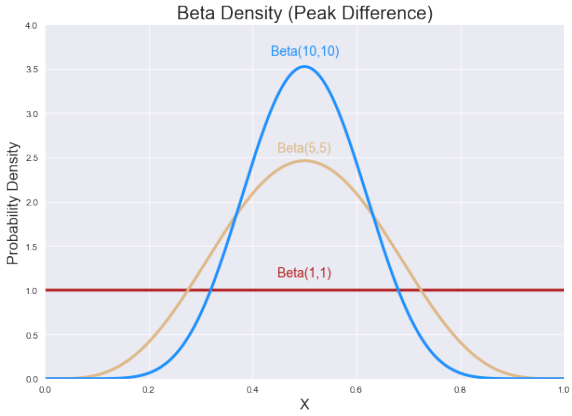

Figure 3: Beta density distribution

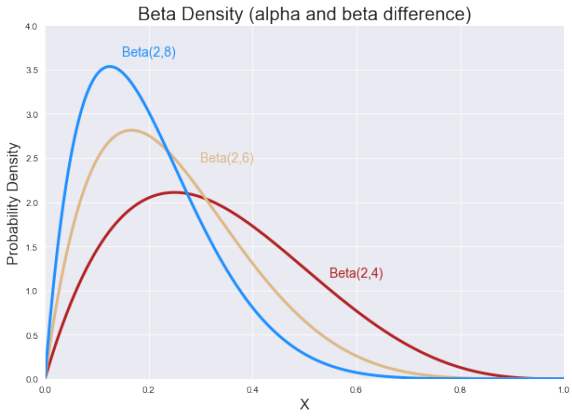

Figure 4: Beta density distribution

Moreover, after having implemented the Thompson Update using the Beta-density, it has been experimentally shown that the larger the amount of Thompson updates the more the distribution peaks around the correct value for p. Plots are presented for several p = 0.2,0.5,0.8 (See Figure 5, 6, 7)

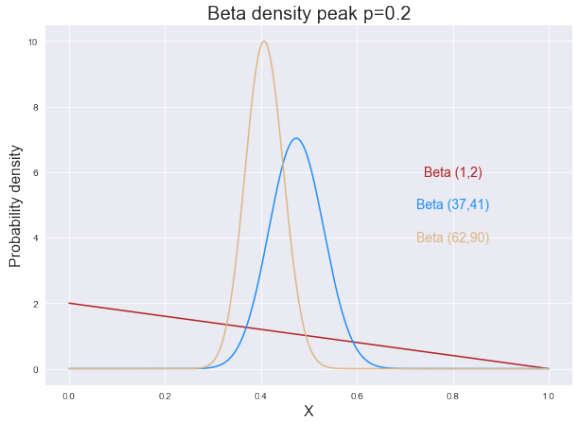

Figure 5: Beta density distribution

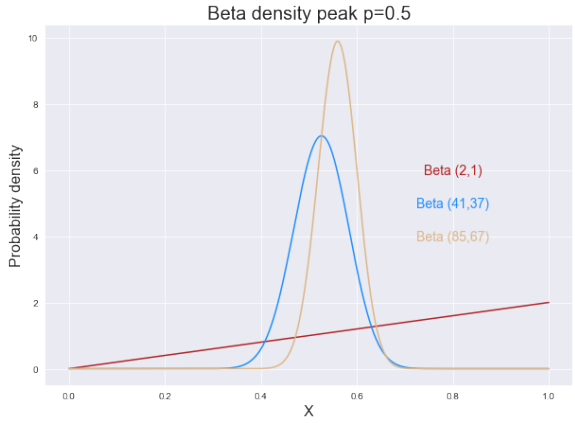

Figure 6: Beta density distribution

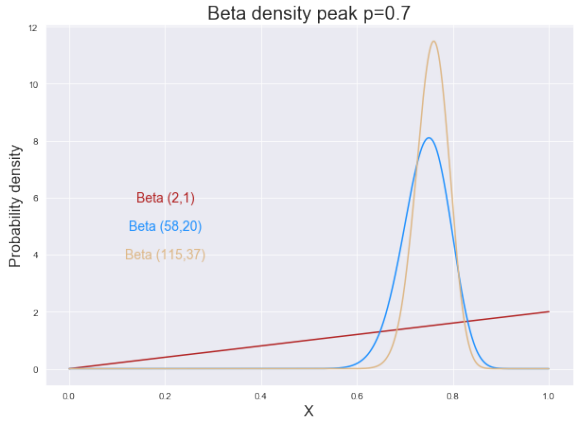

Figure 7: Beta density distribution

Furthermore, the evolution of the mean (See Figure 8) and variance (See Figure 9) for a one arm bandit with $p = 0.5$ over iteration were examined. As visible in the plots . Further, the mean stabilises with time. Similarly, the variance stabilises and decreases to almost 0 with time.

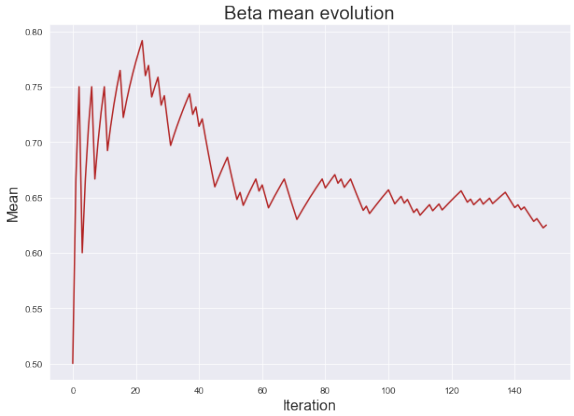

Figure 8: Beta density mean over time

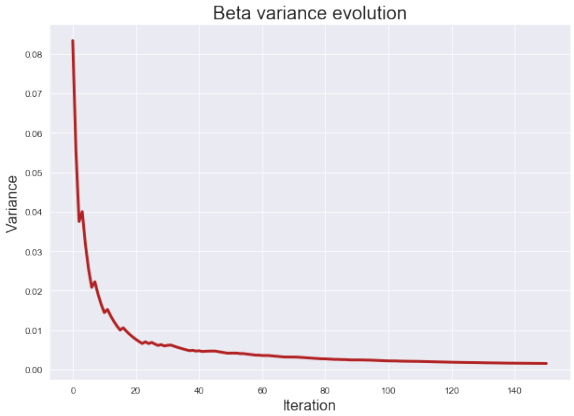

Figure 9: Beta density variance over time

1.2 Thompson sampling for Two Bandit Problem

1.2.1 Implementation

For the Thomspon Sampling as mentioned above, I have decided to use a scipy.stats beta function in order to properly compare different stragies. Similarly to the previous exercise, I was unsure of the uncertainty of the probability value in the bandits. Nonetheless, I decided to act as previously. Probabilities of success were set to $p = 0.4$ in arm 1, and $p = 0.6$ in arm 2. Different values were chosen in order to have a larger difference between the arms, and therefore, allow better evaluation of the performance of the strategies. When implementing the both the UCB and $\epsilon greedy$ function I was using the code from this github account for reference.

1.2.2 Findings

The Thompson Sampling, UCB, and ϵgreedy strategies were compared from several perspectives. For Figures from 11 to 14 a following color assignment continues: ϵgreedy - blue, Thomspon Sampling - red, UCB - beige.

Firstly, the ratio of visiting the arm with higher probability of success (arm 2 with $p = 0.6$) versus visiting the arm with lower probability of success (arm 1 with $p = 0.4$) was examined (See Figure 10). A mean of 100 iterations of all strategies was stored, each with 100 samples within. As seen in the figure, ϵ greedy performed the best, visiting arm 2 95% of the time.

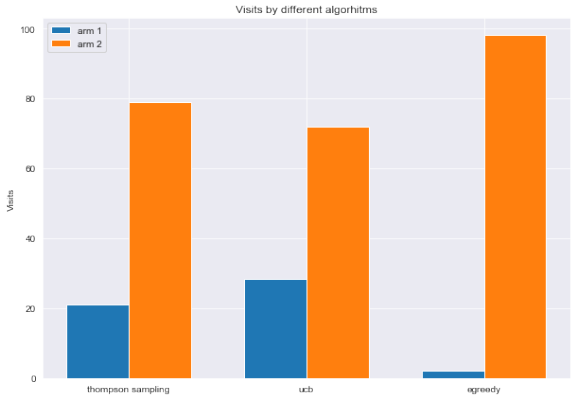

Figure 10: 2 Armed Bandit: Better Arm Mean Visits

Secondly, total reward was examined both with an increasing number of samples and the same $n = 100$ sample number of samples for each iteration of the experiment. As seen in Figure 11 and in Figure 12 the ϵgreedy strategy outperforms the others most of the time. Nonetheless, the Thompson Sampling has a similar performance to the ϵgreedy when looking at the total reward with 100 samples in each iteration.

Thirdly, the average reward was tested. As seen in Figure 13, ϵgreedy surpasses other strategies.

Lastly, the cumulative reward was over-viewed. As seen in Figure 14, ϵgreedy, again, exceeds the other strategies.

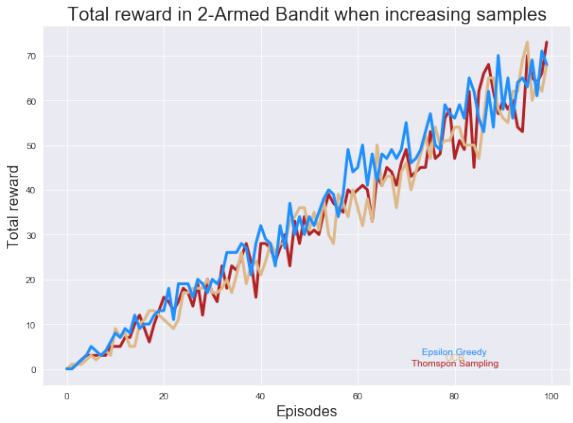

Figure 11: 2 Armed Bandit: Total Reward

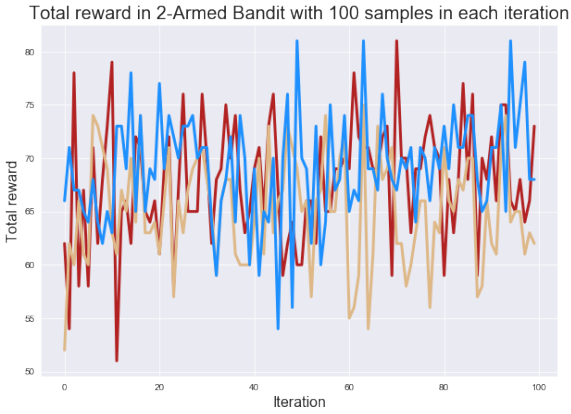

Figure 12: 2 Armed Bandit: Total Reward

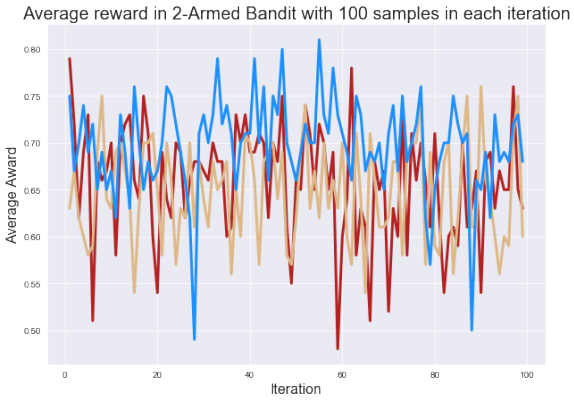

Figure 13: 2 Armed Bandit: Average Reward

2 Reinforcement Learning: Cliff Walking

2.1 Implementation

The cliff grid world was represented by a several lists. The implementation consisted of a series of mutual functions such as transition from a state to state, choosing the next action from the policy and two separate update functions for SARSA and Qlearning. Gamma value has not been used , (equal to set it to 1), to focus mostly on the comparison of performance of the two strategies.

After a series of trials to experiment with the epsilon value, I was unsure if my code was working correctly. Therefore, I have decided to use the code from this github.

Figure 14: 2 Armed Bandit: Cumulative Reward

2.2 Findings

2.2.1 SARSA vs Qlearning

As visible in policies founded by SARSA and Qlearning, SARSA seems to choose a safer approach, whilst Qlearning tends to choose the fastest path to the G (See Figure 15, 16).

```
-------------------------------------------------
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------------
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------------
| R | R | R | R | R | R | R | R | R | R |
-------------------------------------------------
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | R |
-------------------------------------------------
| R | * | * | * | * | * | * | * | * | G |
-------------------------------------------------
```

Figure 15: Policy made by SARSA

```
-------------------------------------------------
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------------
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------------
| 0 | 0 | 0 | 0 | R | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------------
| R | R | R | R | R | R | R | R | R | R |
-------------------------------------------------
| R | * | * | * | * | * | * | * | * | G |
-------------------------------------------------
```

Figure 16: Policy made by Qlearning

This is due to differences in the state value update equation. SARSA algorhitms consider that the next action, because of the the ϵ value or stochastic policy may not be the most beneficial. Therefore, it considers an actual/taken action into account. Qlearning is more optimistic and calculates always the most plausible of the actions from S_{t+1}.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\Big].$$

Source: Introduction to Reinforcement learning by Sutton and Barto — 6.7

Figure 17: Update rule for SARSA

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big].$$

Source: Introduction to Reinforcement learning by Sutton and Barto — 6.8

Figure 18: Update rule for Qlearning

2.2.2 Epsilon value impact

During the experiments with the Epsilon value, an average reward was collected from 10 rounds of both SARSA and Qlearning with 1000 episodes. Unsurprisingly, as seen in Figure 19 and Figure20, the higher the ϵ the lower the average reward, since the actions are more often chosen randomly instead of according to the policy. This differences are even higher for Qlearning, as it normally receives a lower average reward then SARSA.

Figure 19: Epsilon differences for SARSA

Figure 20: Epsilon differences for Qlearning