CS 6955 HW2: Multi-Armed Bandits

Kutay Eken
u1322888

January 2025

1 Part 1:

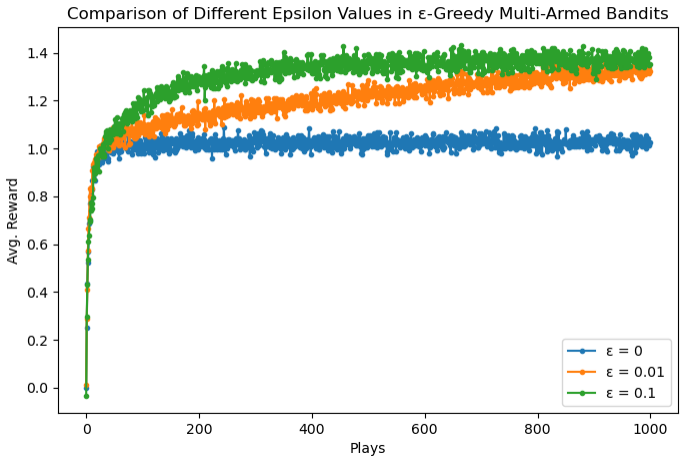

Figure 1: 10-Armed Bandits ϵ-Greedy Comparison for 3 Different ϵ values

Assumptions:
Since I have followed the "Action-Value Methods" section exactly, I expect my graph to closely resemble the Sutton and Barto plot. I created 2000 10-armed bandit agents and implemented the ϵ-Greedy algorithm with three different epsilon values: 0, 0.01, and 0.1. When selecting the greedy action, if multiple actions had the same value, the action with the lowest index was chosen. Each of these 2000 10-armed bandits was run 1000 times for each epsilon value to compute the average reward of the bandit at a given time t.

The initial expected reward for each action was set to 0, and the actual action rewards were randomly assigned using uniform distribution in the range [0,1]. Each action reward was drawn independently from a normal distribution with a mean = actual reward and standard deviation = 1.

In terms of performance, I expect that $\epsilon = 0$ will perform the worst, as it will act purely greedily without exploration. For $\epsilon = 0.01$, I anticipate better performance compared to $\epsilon = 0$. However, while I believe it will eventually converge to the maximum reward, I expect it to do so more slowly than $\epsilon = 0.1$. Therefore, I predict that $\epsilon = 0.1$ will converge the fastest among the three epsilon values and achieve the best performance.

Discussion:
As expected, the graph I obtained closely resembled the Sutton and Barto plot, with $\epsilon = 0.1$ performing the best among the three values. One important consideration is that the number of plays was limited to 1000, and the plot could change with a higher number of plays. If I had tested it with a larger number of plays—which I did not, as the goal was to replicate the Sutton and Barto plot—I would expect $\epsilon = 0.01$ to eventually outperform $\epsilon = 0.1$. This is because $\epsilon = 0.01$ strikes a better balance between random exploration and exploitation. Since $\epsilon = 0.1$ explores less over time, it might perform worse than $\epsilon = 0.01$ in the long run. Overall, all my initial expectations were met.

2 Part 2:

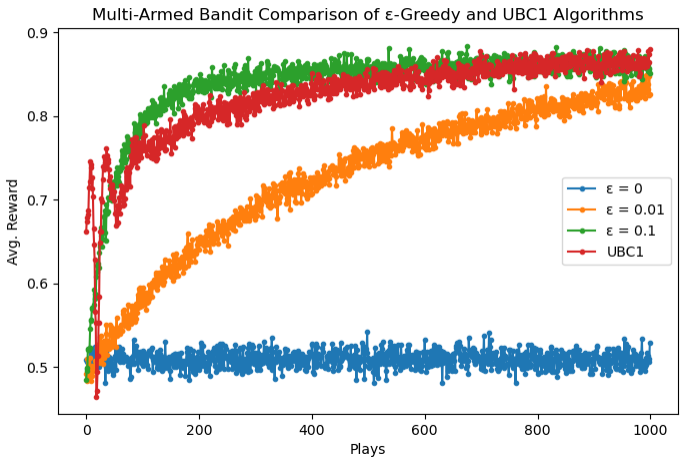

Figure 2: 10-armed Bernoulli Bandits ϵ-Greedy and UBC1 Algorithm Comparison

Assumptions:
For the implementation of the ϵ-Greedy algorithm, everything was kept the same as in Part 1, except for how the action rewards were determined. This time, since it is a Bernoulli Multi-Armed Bandit, a binomial distribution was used to determine the action reward at each time step t. The reward was $+1$ with probability p and 0 with probability $1 - p$. The probability p was randomly selected using a uniform distribution in the range $[0, 1]$.

For the UCB1 part of the implementation, there were some major differences. The initial reward estimate for each action was determined by pulling each action once, and actions were selected at each timestep using the following formula:

$$A_t = \arg\max_a \left(Q_t(a) + \sqrt{\frac{\ln(t)}{N_t(a)}} \right)$$

Where:

- t = timesteps
- $N_t(a)$ = number of times action a is taken
- $Q_t(a)$ = estimated value of action a

If multiple actions satisfied the condition, the action with the smallest index was chosen.

In terms of performance, my expectations for the ϵ-Greedy algorithm remain the same as in Part 1. For the UCB1 algorithm, I expect it to perform the best. While I do not anticipate it significantly outperforming the $\epsilon = 0.1$ strategy (since it almost maximized the reward previously), I expect it to converge faster.

Discussion:
As can be seen from the graph, the ϵ-Greedy plot remained largely unchanged, while UCB1 converged faster. However, the graph also shows that the average reward values for UCB1 fluctuated at the beginning. As the agent played more, the average reward values became more consistent and increased over time.

One important consideration is that when multiple actions were available for selection, I always chose the action with the smallest index. The graph, particularly the UCB1 part, could have looked different if I had used an alternative strategy, such as randomly selecting from the available actions. This approach could have allowed the agent to explore other actions more effectively and potentially learn better.

Overall, my previous assumptions and expectations were correct.