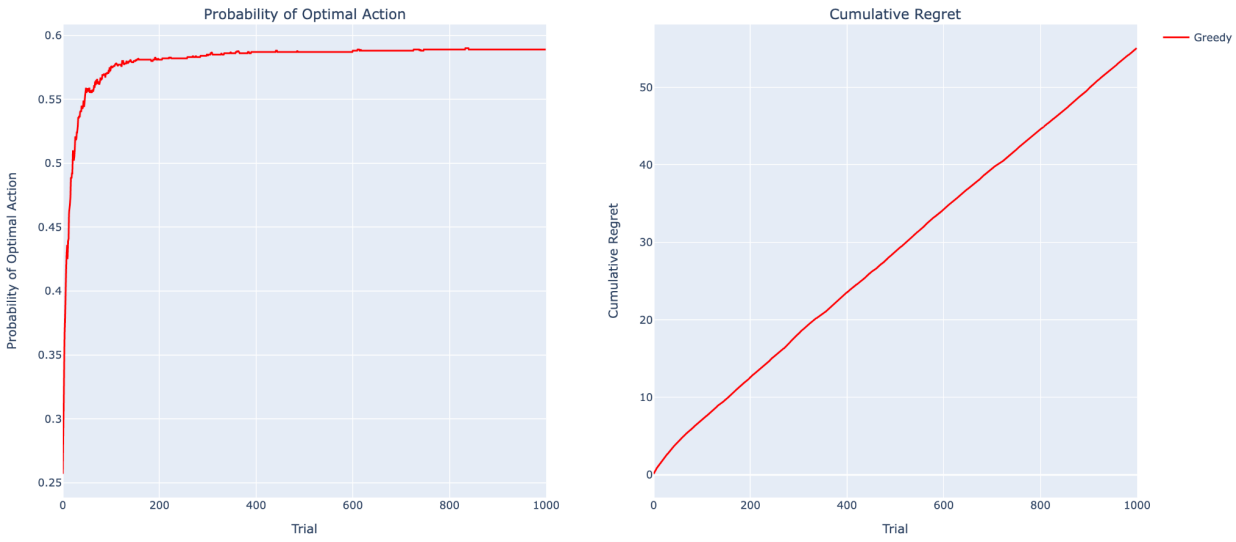
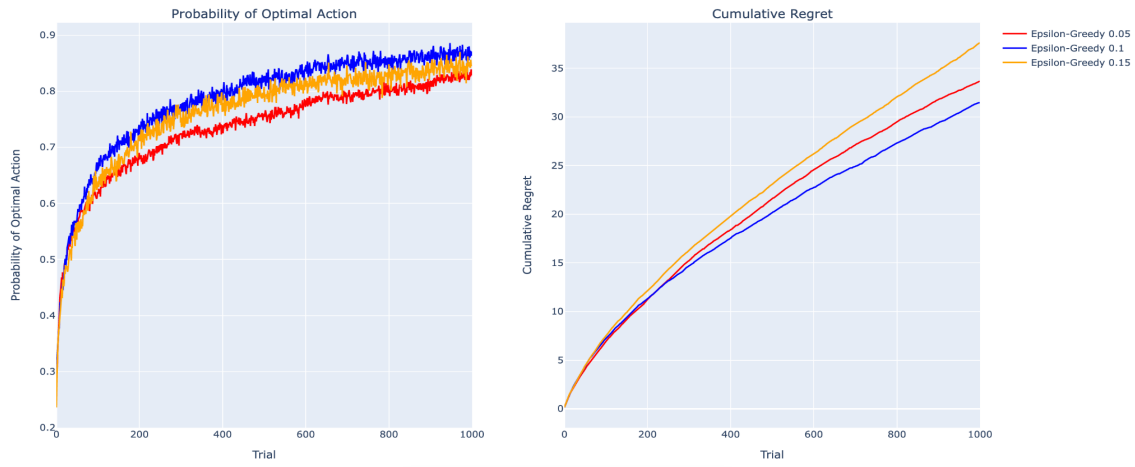
Zander Zemliak
CMSI 4320
February 3, 2021

# CMSI 4320 Assignment #1

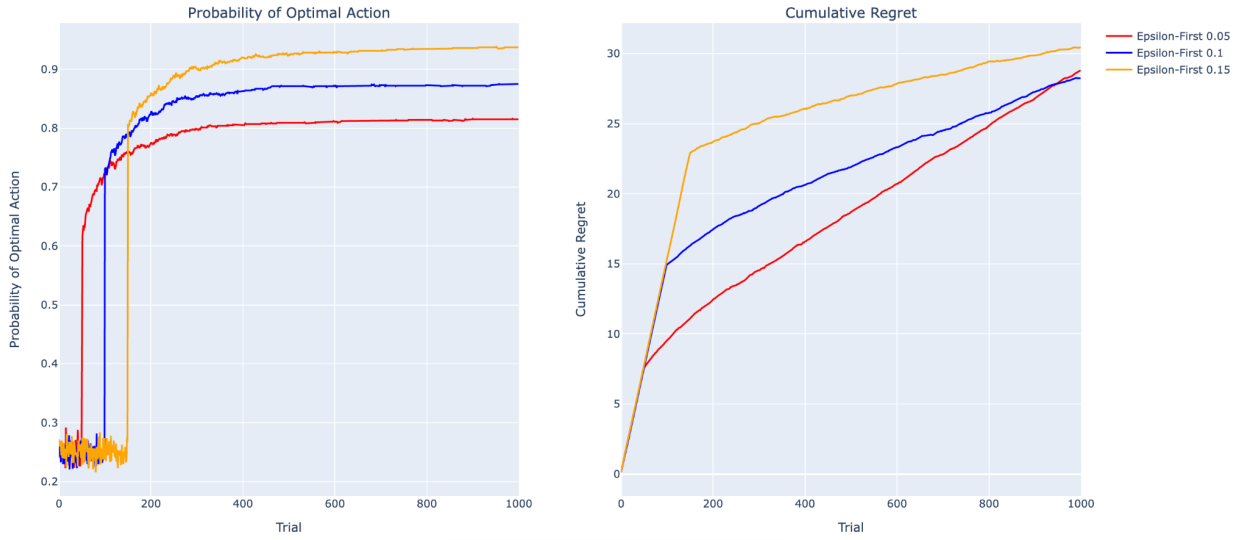
1. I do not expect this agent to perform well because the agent could get unlucky with what is the overall best action at the start of its experiment and then never choose the actual best action ever again in the future.




2. I expect the Greedy-Epsilon agent to perform slightly better than the purely Greedy Agent due to its possibility of adjusting its history towards the better action in the event we get stuck exploiting a good action, but not the best. I think that the higher epsilon value of 0.15 will perform the best because it is more likely the agent will adjust towards the "better" action.



3. I am expecting the Epsilon-First agent to perform better than Epsilon-Greedy because it allows the agent to explore randomly what could be the best action after a certain amount of time and then it is likely that we will have adjusted towards the more "right" action and then by using the greedy approach after that set amount of time, the best action will become more prevalent.




4. For the Epsilon-Decreasing agent, I am attempting three different functions. The three functions are a rapid exponential decay function, a linear function, and a slow decaying decaying function. The functions look like this: $epsilon = prior\ epsilon - 0.001$, $epsilon = e^{-t+2}$, and $epsilon = \frac{1}{(t+45)^{\frac{1}{2}}}$. The best performing function was the slow decaying function (the third one stated).

5. Ran out of time to figure it out :( (I am going to start doing these assignments earlier).

6. I did a combination of Epsilon First but instead of using the purely Greedy algorithm, I used the Upper-Confidence-Bound algorithm instead with a low c parameter since I already did my exploration in the beginning. The procedure looks like this:
    a. Conduct T * epsilon trials randomly picking arms
    b. Conduct the remaining T - (T * epsilon) trials using the UCB algorithm

    $$argmax_a = (Q_t(a) + c * \sqrt{\frac{log(t)}{N_t(a)}})$$

    I do not know if it was luck, but my custom procedure turned out to perform the best out every one


7. POKER (Price of Knowledge and Estimated Rewards) is a strategy that attempts to balance exploration and exploitation. This algorithm relies on pricing uncertainty, exploiting the lever distribution, and time. The way this method differs from the traditional MAB algorithms is that POKER attempts to quantify uncertainty and make decisions based on that result. At one point, once a certain number of trials remain, the strategy switches to purely exploitation. This algorithm is very good at dealing with a problem known as the Content Distribution Network problem. With many pathways, the objective is to pick the one which minimizes latency.