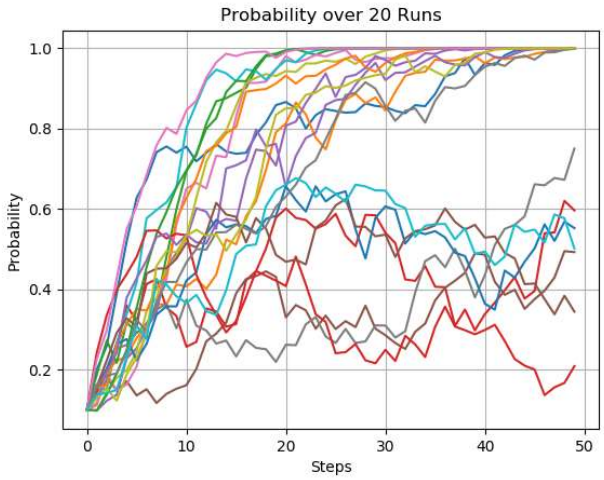
Name:
Nikolas Maier
Oluwatomilayo Adegbite

Student #:
500461990
500569283

Report on L_{R-P} and L_{R-I}

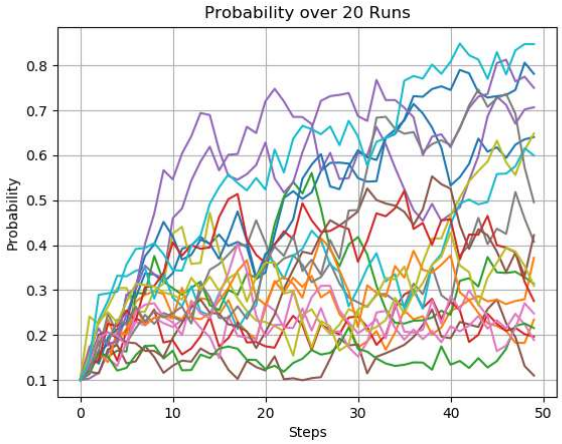
After creating and running both L_{R-I} and L_{R-P} the results were rather fascinating. L_{R-I} trended towards picking the optimal arm more frequently, with some runs leading to picking the optimal arm by the 5000th almost 100% of the time. If you look at the below graph, you can see a small subset of only 20 runs to make graphing it more legible. Of the 20, 7 had results below 80% chance of picking the optimal arm while 13 had either 99-100% chances of picking it.



L_{R-I} Algorithms probability of picking optimal arm (Steps are actually for every Thousand)

This trend lead to an overall average of 86.3% chance over the 100 iterations of picking the optimal arm out of all 10. The L_{R-I} algorithm also averaged picking the Optimal arm 3351.95 out of 5000, with an average reward score of 4317.76.

The good results of L_{R-I} are in contrast to the starker results of L_{R-P}, which had a more dismal turnout on average. The below graph should give an idea on the lesser results. Of the 20 that were run only 4 got above a 70% chance of picking the optimal arm.



L_{R-P} Algorithms probability of picking optimal arm (Steps are actually for every Thousand)

Overall the averages of L_{R-P} show a rather large difference compared to the previous algorithm. L_{R-P} on average will end with a probability of picking the optimal arm of around 40%. It had an average reward of only 3519.68. This result is almost 1000 less points than L_{R-I}. The reasoning we found for why the Penalty algorithm gives such worse results than Inaction is that a string of bad luck could penalize a slot arm that is the optimal arm into such low probability of being picked again that it is irrelevant. With Inaction, it would take a string of good luck on another arm's results to lower the optimal arms numbers, and in that scenario it would generally mean another good arm has been found. The punishment due to randomness might have been too heavy in regards to L_{R-P}.

These algorithms in comparison to UCB gives us some interesting insights. While UCB is quite good at achieving high reward, it isn't always very accurate at picking the optimal arm. UCB tends to deviate between either picking the optimal arm almost 100% of the time or not at all. The Learning algorithms tend to slowly pick and choose, and in the Reward-Inaction algorithm, heavily end up favouring the Optimal Arm by the end of the 5000 pulls. What this means is that L_{R-I} and UCB tend to gather as much rewards as each other even if UCB doesn't identify the optimal arm on its current run. If gathering rewards is the goal than either algorithm should work, though emphasis on UCB seems deserved. If finding the optimal arm is what is required, then L_{R-I} should be the algorithm used.

The results mentioned about averages were gathered in a dataset generated by the python code. It has also been included in a spreadsheet placed in the zip folder if you wish to look over the results.