NAME: Galen Byrd
DATE: 02/02/19
Homework: HW3
STAT/CS 387

Problem 1

I set up my Multi-armed Bandits in the same way that was shown in MAB.html (lines 115-138). I decided to proceed using the same five beta distributions as the example, as the means of the distributions are spaced out between [0,1] and there is a decent amount of overlap for potential rewards from different bandits, see Fig 1. This means this distribution is an average difficulty MAB problem, as reward distributions with no overlap are easy to solve while reward distributions that overlap a lot and differ only slightly in means are harder to optimize.

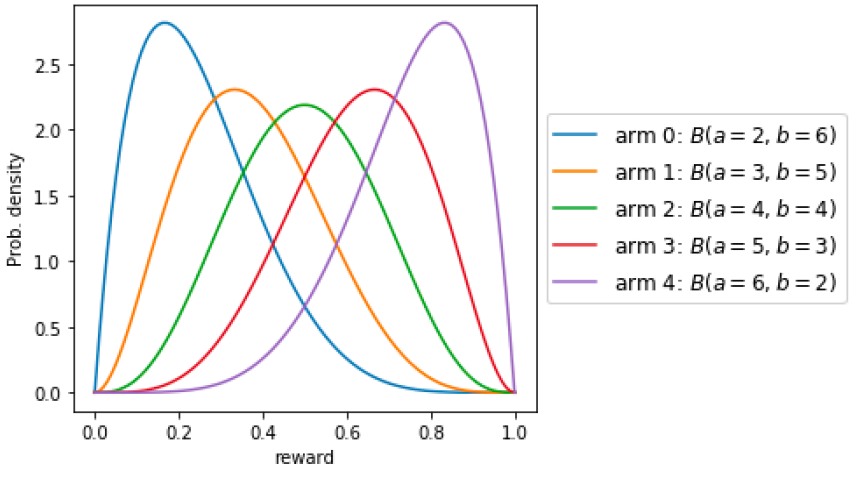

Fig 1: True distribution of rewards for arms

Problem 2

I proceeded implementing the four algorithms. The random approach is simply pulling a random arm each time (lines 24-32). The naive greedy approach includes initializing our estimate for each arm at 0.5, as we are unsure of any estimates. We immediately go forward with pulling our highest estimate, updating our estimated each time-step. This means the pure greedy algorithm can get stuck in a sub-optimal arm extremely easily if it never even gets a chance to pull the best arm (lines 34-47). Epsilon first greedy is different in that it has an exploration and

an exploitation phase (lines 50-75). I included the number of cycles of exploration as a parameter to my function so it can be tuned as seen fit. Lastly, for epsilon greedy we randomly play an initial arm, followed by playing the arm with the largest estimated reward with Pr(1-epsilon) and playing a random arm otherwise (lines 77-95). I included epsilon as a parameter to the function again so this can be tuned as seen fit. This algorithm is unlikely to get stuck at local optima, but incurs high regret over long gambles as arms are randomly picked every so often. I ensured these strategies are not cheating by not allowing the gambler to see the true reward distribution of the arms. I do use the true max expected reward and the knowledge that arm 4 is the best choice in calculating regret and my metric, but this is ok as it is evaluating the performance of the algorithm, not adding information for the gambler.

Problem 3

I decided to use the percent of pulls in a gamble that were of the optimal arm as my second performance metric. This metric applies to all of the algorithms, as some do not converge to the optimal arm ever so time to convergence would not work well as a metric. This percent is also informative, as ideally we would maximize this percent to 100 if we knew the true reward distributions for the arms. I set up easy and hard bandits using the beta distribution, making sure there was no overlap in distributions for the easy bandit and a lot of overlap for the hard bandit, see Fig 2 and Fig 3 (Lines 269-295).

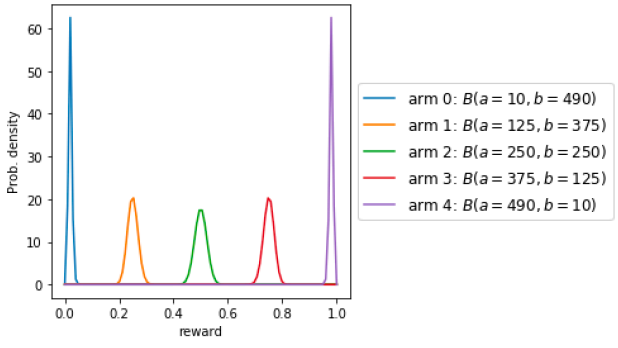

Fig 2: Reward distributions for easy bandits

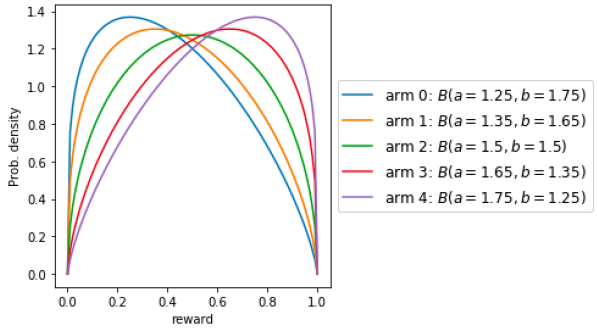

Fig 3: Reward distributions for hard bandits

I proceed to evaluate the four algorithms discussed in class using all three distributions of rewards. I set up my code so that I could test the algorithms individually while I was working on implementing them (Lines 142-217) and run them all for my final output (Lines 221-264). I use epsilon=.85 for epsilon first greedy and m=4 for epsilon greedy for all evaluations. The following figures show the regret of one gamble over time, averaged over 100 gambles, and the

percent of optimal pulls of a strategy over time. Figures 4-6 were evaluated using the average difficulty bandits, Figures 7-9 using the easy bandits and Figures 10-12 using the hard bandits.

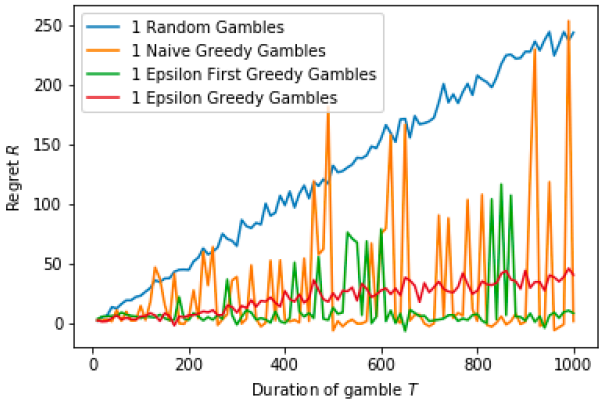

Fig 4: Regret of one gamble over time

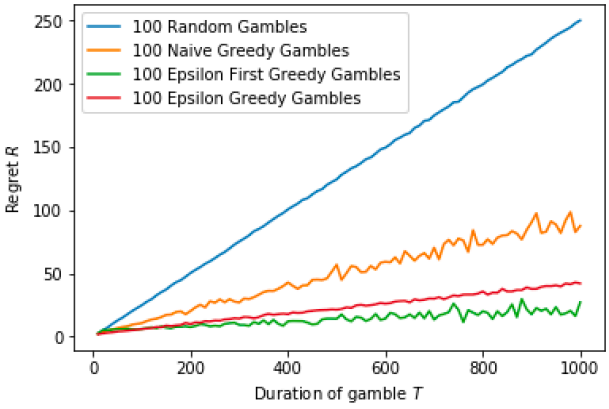

Fig 5: Regret of 100 gambles over time

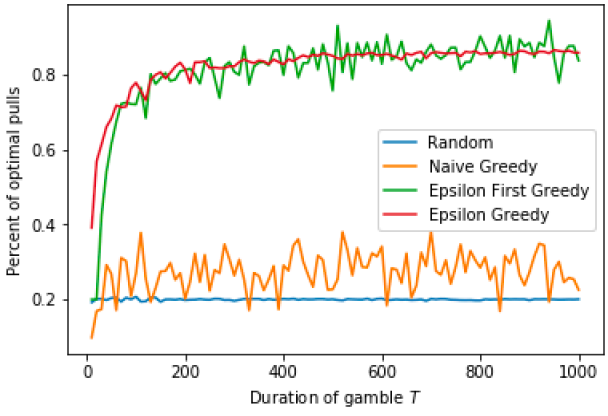

Fig 6: Percent of optimal pulls averaged over 100 gambles over time

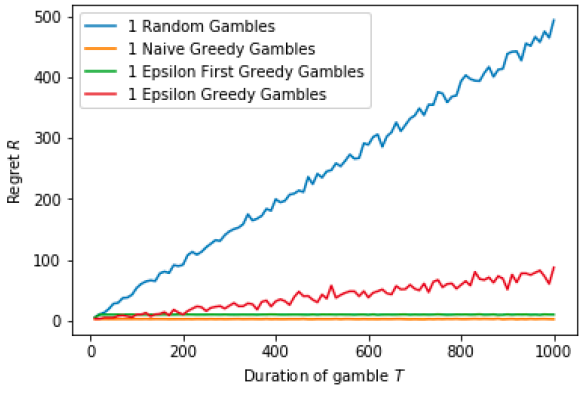

Fig 7: Regret of one gamble over time

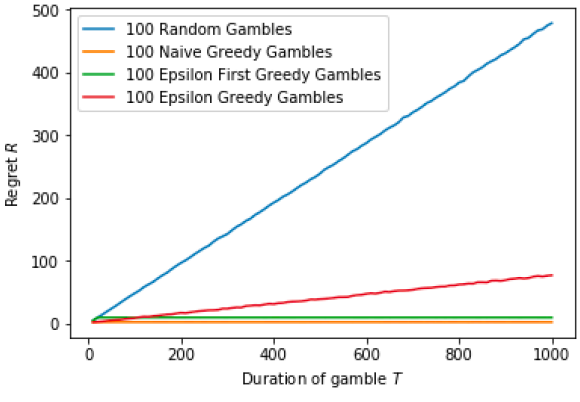

Fig 8: Regret of 100 gambles over time

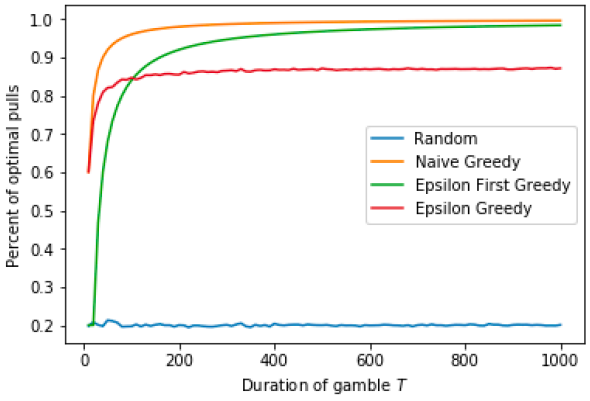

Fig 9: Percent of optimal pulls averaged over 100 gambles over time

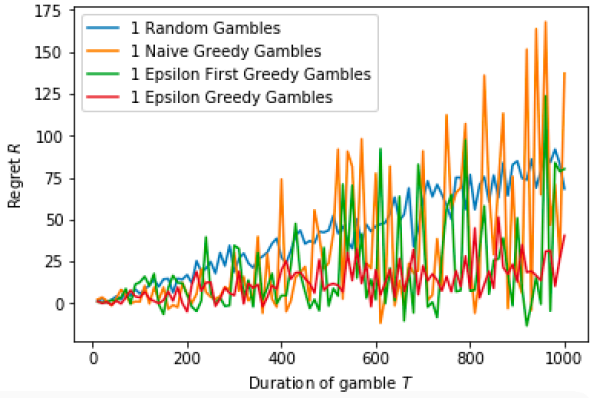

Fig 10: Regret of one gamble over time

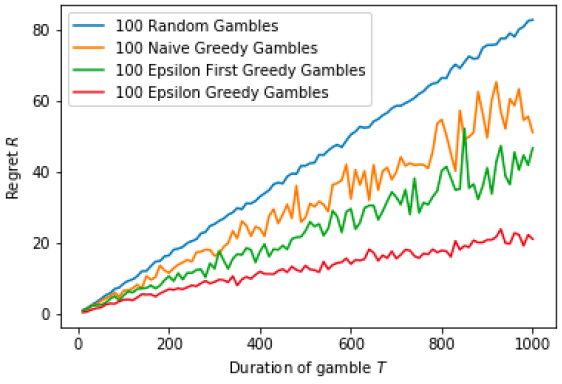

Fig 11: Regret of 100 gambles over time

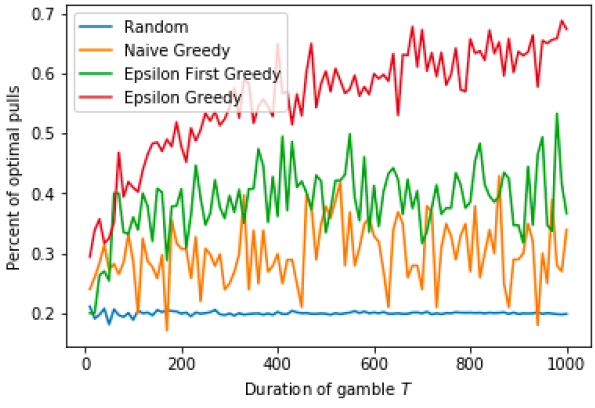

Fig 12: Percent of optimal pulls averaged over 100 gambles over time

For the average difficulty bandits we see the lowest regret from the epsilon first greedy algorithm, for easy bandits the lowest regret is pure greedy, followed closely by epsilon first

greedy, and for the hard bandits epsilon greedy gives us the lowest regret over time. I propose that, since the gambler cannot know the difficulty of the distribution of rewards, the best algorithm to solve any difficulty level multi-armed bandit would be the epsilon greedy algorithm. Even though it does not perform as well as pure greedy and epsilon first greedy for the easy bandits, because of the possibility of random pulls, it has the advantage of never getting trapped pulling a sub-optimal arm. These other algorithms have the possibility of getting stuck in a sub-optimal arm after their exploration period as they do not update the reward estimates in the exploitation period. This comes back to haunt these strategies in solving a difficult MAB. We also see that the epsilon greedy algorithm converges to its max percent of optimal pulls early on in the gamble for the easy bandits and first for the average and hard bandits. While its max value is not close to 1 because of the random pulls, it does reach its max early on in a gamble, therefore the regret of the entire gamble is unlikely to spike.

UCB1 Performance

I proceed comparing the UCB1 algorithm to both the epsilon greedy algorithm and the epsilon first greedy algorithm on average (Figs 13-15), easy (Figs 16-18) and hard (Figs 19-21) bandits.

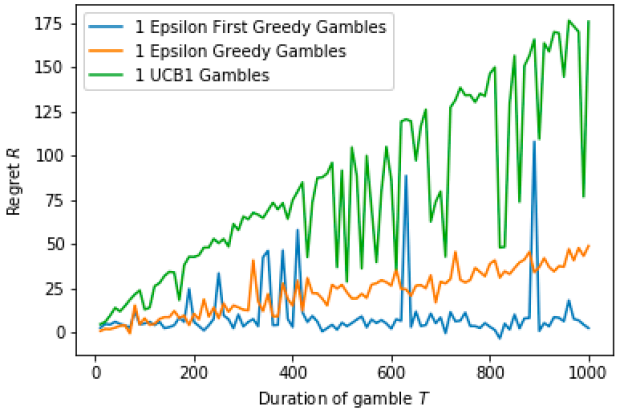

Fig 13: Regret of one gamble over time

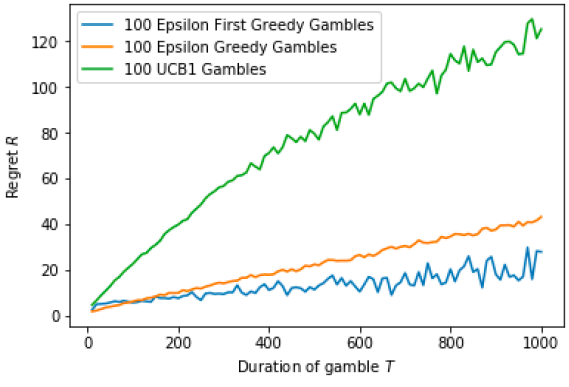

Fig 14: Regret of 100 gambles over time

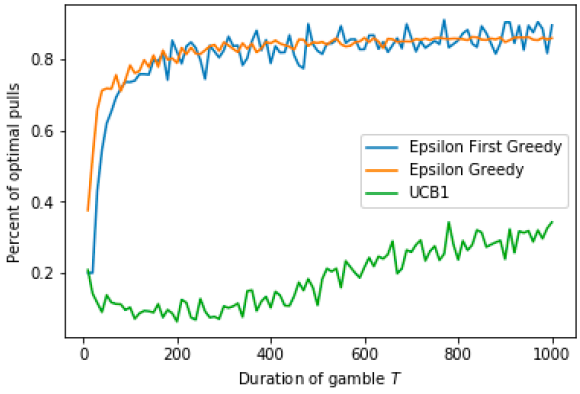

Fig 15: Percent of optimal pulls averaged over 100 gambles over time

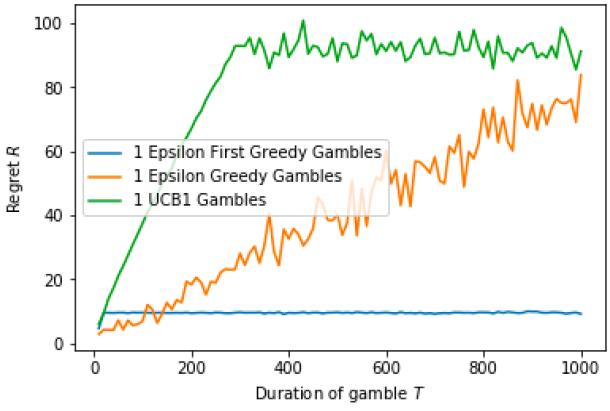

Fig 16: Regret of one gamble over time

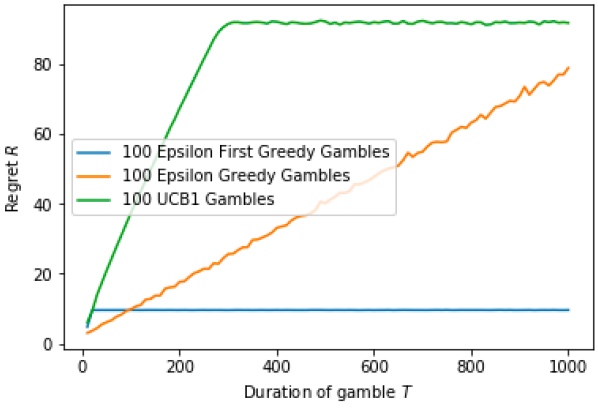

Fig 17: Regret of 100 gambles over time

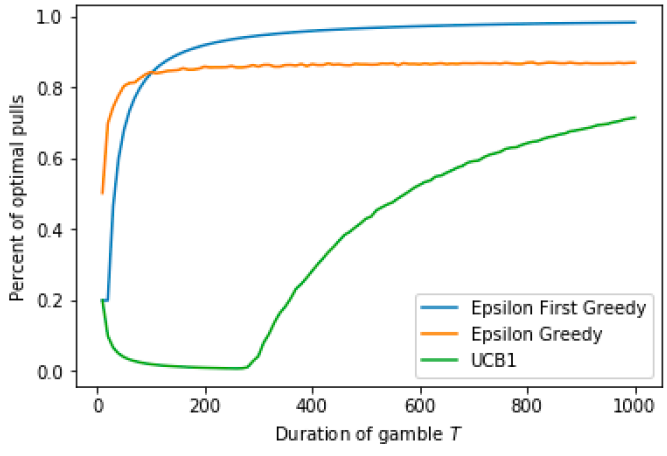

Fig 18: Percent of optimal pulls averaged over 100 gambles over time

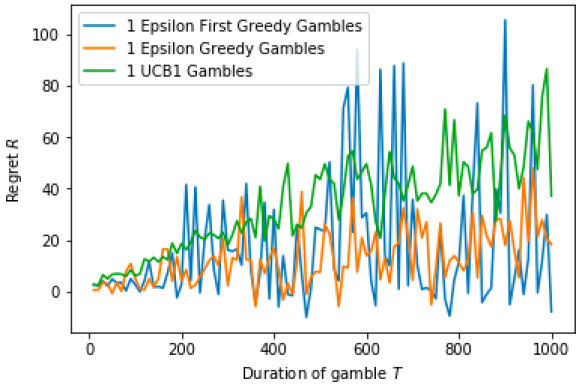

Fig 19: Regret of one gamble over time

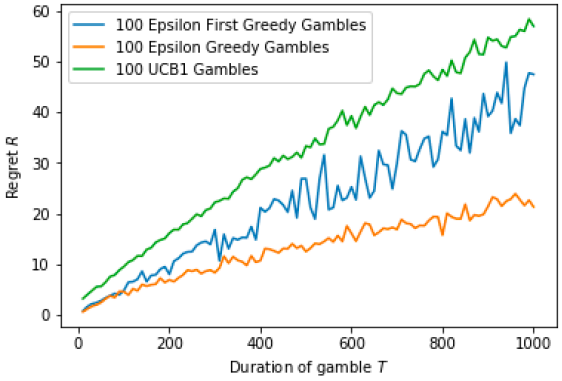

Fig 20: Regret of 100 gambles over time

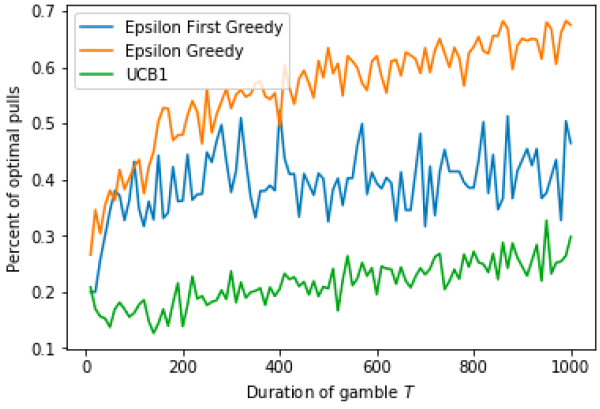

Fig 21: Percent of optimal pulls averaged over 100 gambles over time

We can see that my UCB1 algorithm gets stuck in a sub-optimal arm for a long time early on before finding the optimal arm. I believe this happens because the term weighting uncertainty in the algorithm remains high while it is early on in the gamble, although I am not certain. My implementation of the UCB1 algorithm (Lines 97-111) is very similar to the epsilon first algorithm, except that it uses a different estimator than simply the expected reward, in trying to account for uncertainty. Again, through my analysis if I were the gambler I would have the most confidence in the epsilon greedy algorithm even though the epsilon first greedy algorithm does so well for the easy bandits, and theoretically UCB1 should be the best.