# Assignment 1 Report

Sriram Vijendran

03/02/20

## 1 Solution to Q1 ($\epsilon$-greedy)

- The 10-arm testbed is implemented with each arm being a standard normal distribution where the mean value of each arm was randomly selected from a Gaussian distribution. For the epsilon greedy algorithm we run the agent through the testbed for 1000 iterations where the agent could "pull the arms" of the bandit 2000 times before the environment was reset.

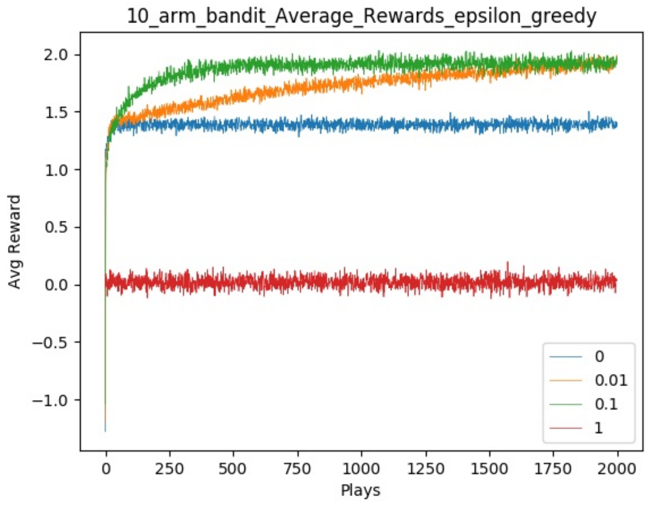

Figure 1: 10 Arm Epsilon Greedy Average Rewards

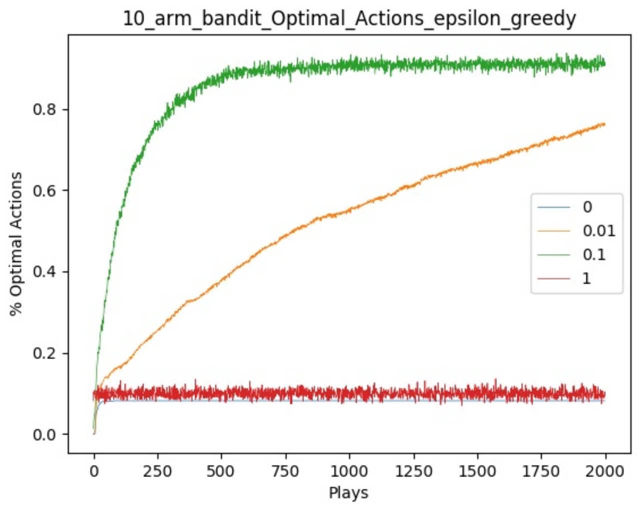

Figure 2: 10 Arm Epsilon Greedy Optimal Actions

- For the epsilon greedy agent we compare 4 different values of epsilon[0, 0.01, 0.1, 1]. Figure 1 shows the average rewards earned by each of the bandits. It is clear that for $\epsilon$=0 the agent converged to a higher average reward of a little under 1.5 per play faster than for other values of $\epsilon$, but does not improve from there. $\epsilon$=1 does no exploration and never improves from 0 rewards. $\epsilon$=0.01 shows a sharp increase to an average reward of 1.5 then makes a steady climb to reward 2 per play. $\epsilon$=0.1 makes a direct jump to reward of 2 per play and shows the best performance, utilizing both exploration and exploitation.

- As expected from studying the Average rewards from Figure 1, the % of optimal actions follow the same pattern where $\epsilon$=0.1 converges faster than other values of $\epsilon$, while $\epsilon$=0 and $\epsilon$=1 never improve.

## 2 Solution to Q2

- Softmax action selection is also implemented is the same manner i.e. 2000 plays over 1000 iterations, and the average reward earned and % of optimal actions are plotted.
- Softmax action selection is used with three different temperatures [0.1, 1, 0.01]. The softmax action selection selects each arm with a probability drawn from a Gibbs(Boltzmann) distribution using the formula:

$$p_i(t+1) = \frac{e^{q_i/t}}{\sum_{j=1}^{k} e^{q_i/t}}, i = 1, ....k$$

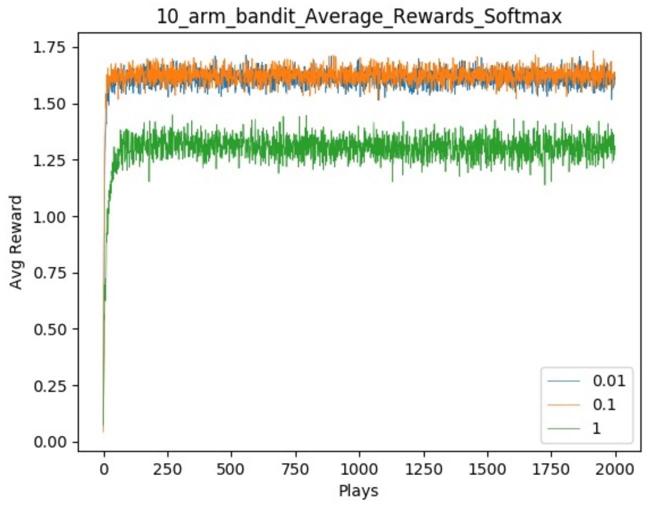

Figure 3: 10 Arm Softmax action Average Rewards

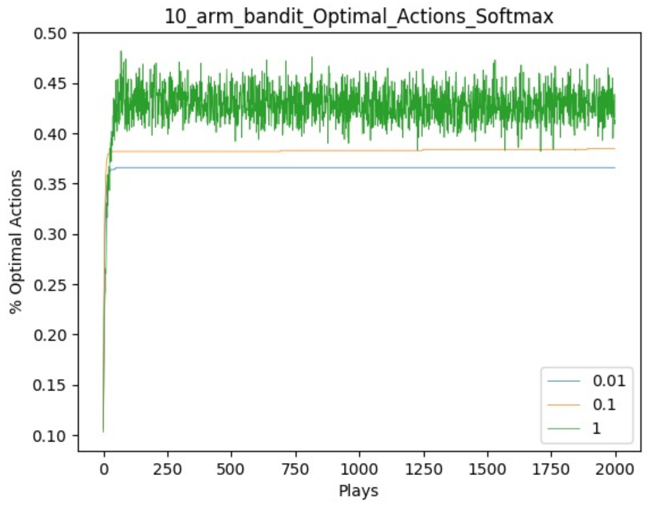

Figure 4: 10 Arm Softmax Action Optimal Actions

- The temperature controls the "greediness" of the agent. Temperatures close to 0 will act extremely greedy while temperatures tending to infinity will act more "explorative". As shown in the graph, the temperature of 0.1 and 0.01 achieves the highest average rewards, while picking the less optimal actions more often, than a temperature of 1 that picks the actions most evenly.

## 3 Solution to Q3 UCB1 algorithm

- UCB1 is also implemented in a similar manner to the previous 2 algorithms.

- UCB1 selects the arms based on their potential to yield a high reward. It begins by sampling each arm at least once, the starts to sample arms based on the algorithm:

$$A_t = \underset{x}{\text{argmax}}[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}]$$

where $N_t(a)$ refers to the number of times that particular arm was pulled prior to time $t$, $Q_t(a)$ is the estimated value of the arm a, and

$c > 0$ is a parameter that controls the explorative/exploitative nature of the algorithm.

- The square root term controls the variance of the expected reward of a particular arm, and the denominator of the square root term ensures that the variance of expected returns of that arm reduces with eachsampling of that arm.

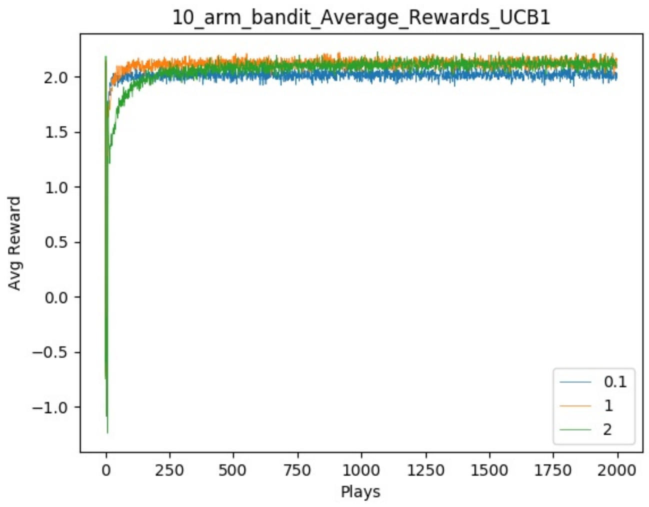

Figure 5: 10 Arm UCB1 action Average Rewards

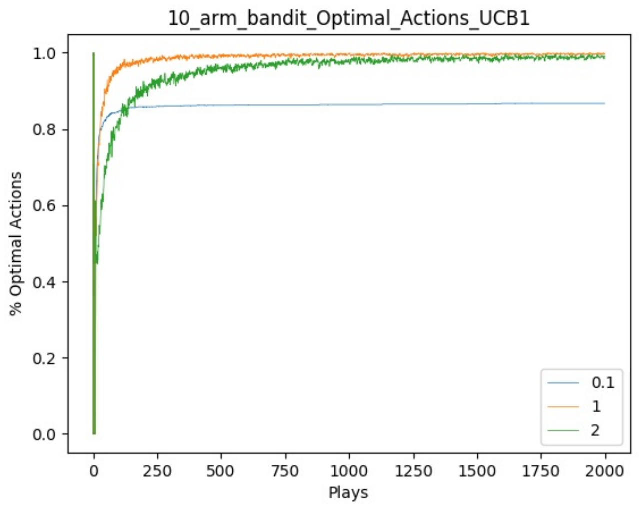

Figure 6: 10 Arm UCB1 Action Optimal Actions

- As shown in Figure 5, the UCB1 algorithm quickly converges to the maximum value and begins to exploit the most optimal arm early on. There is an initial sharp increase the average rewards, which then drops and again steady surpasses that value. This is due to the UCB1 algorithm initially exploiting a sub-optimal arm as it has higher expected return. Soon the variance of that particular arm drops and the algorithm exploits another arm which has better returns

- this process of exploiting the arm with maximum expected returns while reducing the variance of future returns ensures that the UCB1 algorithm will eventually converge on the most optimal arm.

- We continue to compare the performance of the UCB1 algorithm to $\epsilon$-greedy and softmax action selection. It is clear that UCB1 successfully exploits the most optimal arm faster than $\epsilon$-greedy and softmax action selection, yielding the most rewards.

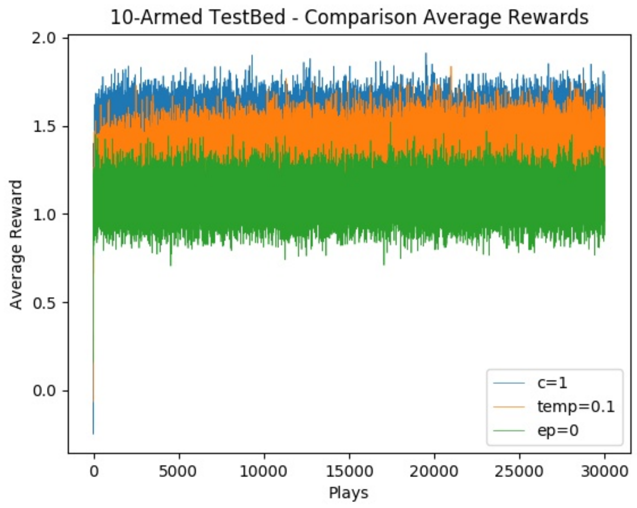

Figure 7: 10 Arm Comparison action Average Rewards

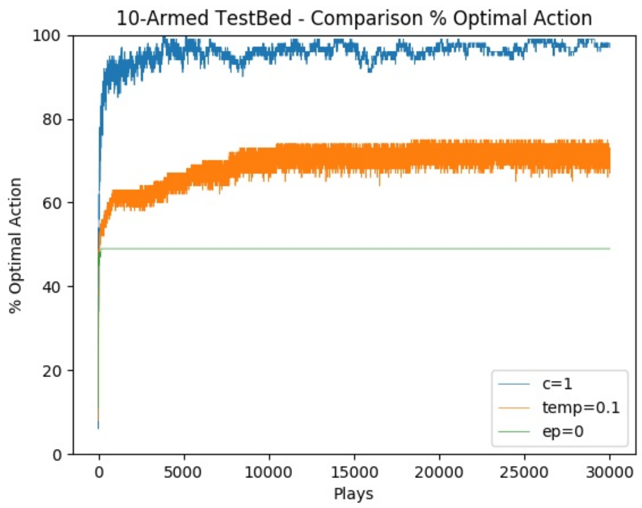

Figure 8: 10 Arm Comparison Action Optimal Actions

- The reason for the better performance of UCB1 over the other two is due to the selection of the values of $\epsilon$ and temperature. The performance of $\epsilon$-greedy and softmax selection is purely dependent on the the selection of the ideal values of $\epsilon$ and temperature. UCB1 does not require such selection of parameter. It will converge on the most optimal arm regardless of the value of c.

## 4 Solution to Q4 MEA algorithm

- The median elimination algorithm is a type of PAC optimality algorithm, also called ($\epsilon$, $\delta$)-PAC algorithm. It samples all the arms of the bandit a fixed number of times given by the equation:

$$n_l = (\frac{1}{(\epsilon_l/2)^2}) \log(\frac{3}{\delta_l})$$

  Where $\epsilon_l = \epsilon \times \frac{1}{4}$ and $\delta_l = \delta \times \frac{1}{2}$. After each round of sampling, half the bad arms are eliminated by removing the arms whose expected values are less than the median of the expected values.

- After each round of elimination, the values of $\epsilon_l$ and $\delta_l$ are updated according to:

$$\epsilon_{l+1} = \epsilon_l \times \frac{3}{4}, \delta_{l+1} = \delta_1 \times \frac{1}{2}$$

- From the graphs below it is clear that large values of $\epsilon$ use fewer number of samples to converge at the optimal arm since we don't need to be confident about the arms, while larger values of epsilon require more samples as they need to be more confident before elimination.

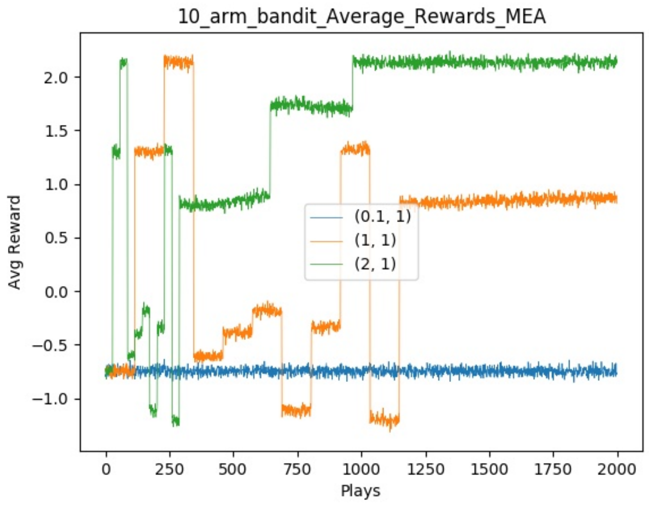

Figure 9: 10 Arm MEA action Average Rewards

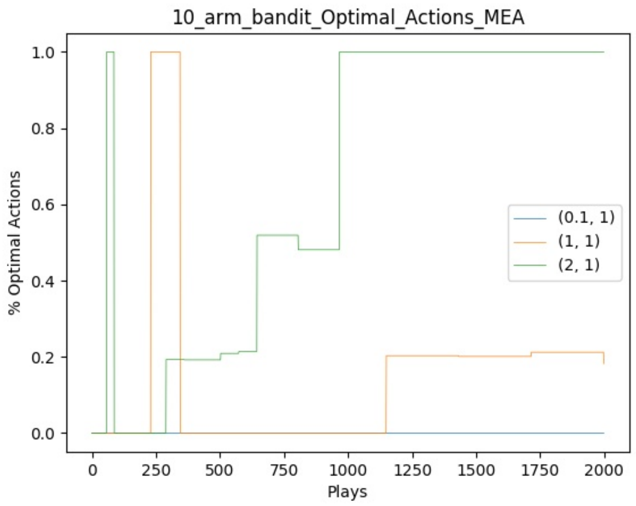

Figure 10: 10 Arm MEA Action Optimal Actions

- The rate determining step in Median Elimination is not the calculation of the median in problems with only a few arms i.e. 10 arms. Median elimination has a complexity of $O(k \log k)$ while the other algorithms so far have a complexity of $O(k)$, which does not make much difference for small number of arms. However when dealing with many more arms i.e. 1000 arms, the Median Elimination algorithm performs better and the calculation of the median is the rate determining step.

## 5 Solution to Q5 100 arms testbed

- We now compare the performance of the aforementioned algorithms on a 1000 arm testbed. The simulation consists of a 1000 arm bandit where the algorithms run for 100 iterations with 30,000 plays each iteration.
- It can be seen from the follow graphs that the UCB1 algorithm converges to the peak value and obtains maximum rewards very quickly. It has the best performance compared to both softmax and $\epsilon$-greedy.

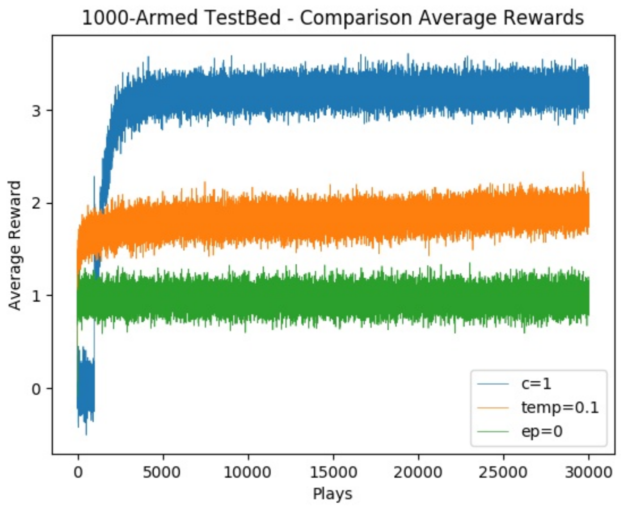

Figure 11: 1000 Arm comparison

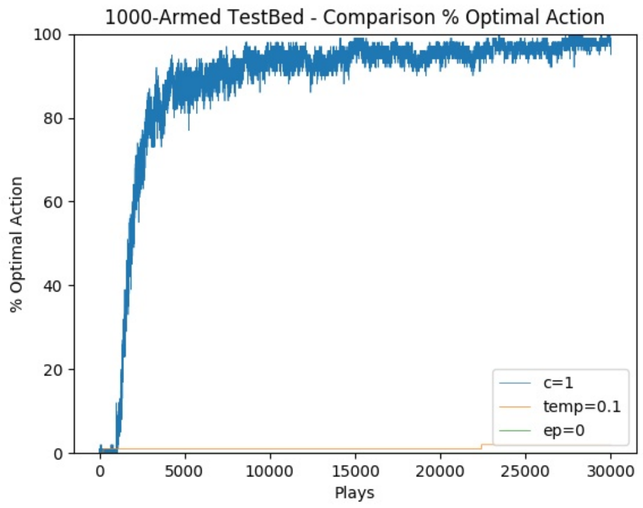

Figure 12: 1000 Arm comparison

- From the optimal actions taken it can be seen that both softmax selection and $\epsilon$-greedy are not able to select the more optimal arms as frequently as UCB1.
- Initially, the UCB1 algorithm does not give a good performance, but with time it is able to move from selecting sub-optimal arms to optimal arms. the reason for this is that $N_t(a)$ is very low and it has not sampled each arm sufficiently to properly estimate the return for that arm.
- to improve the performance of $\epsilon$-greedy and softmax can be improved by setting the right $\epsilon$ and temperature.