Reinforcement Learning - CS6700
Assignment - 1

K Pawan Prasad, **Roll Number: ME16B179**

February 2020

Contents

1 NOTE-1

For Questions 1 to 3

- Number of arms (k) = 10
- Number of time steps = 1000
- Number of runs = 2000

2 Question 1

- Algorithm: ϵ-Greedy

Figure 1 shows the learning curve for different ϵ parameter values. We have taken ϵ values = [0,0.01,0.1,0.2].The arm with highest expectation is picked with a probability of (1-ϵ+ϵ/k) and the any of the other arms are selected with probability of (ϵ/k)

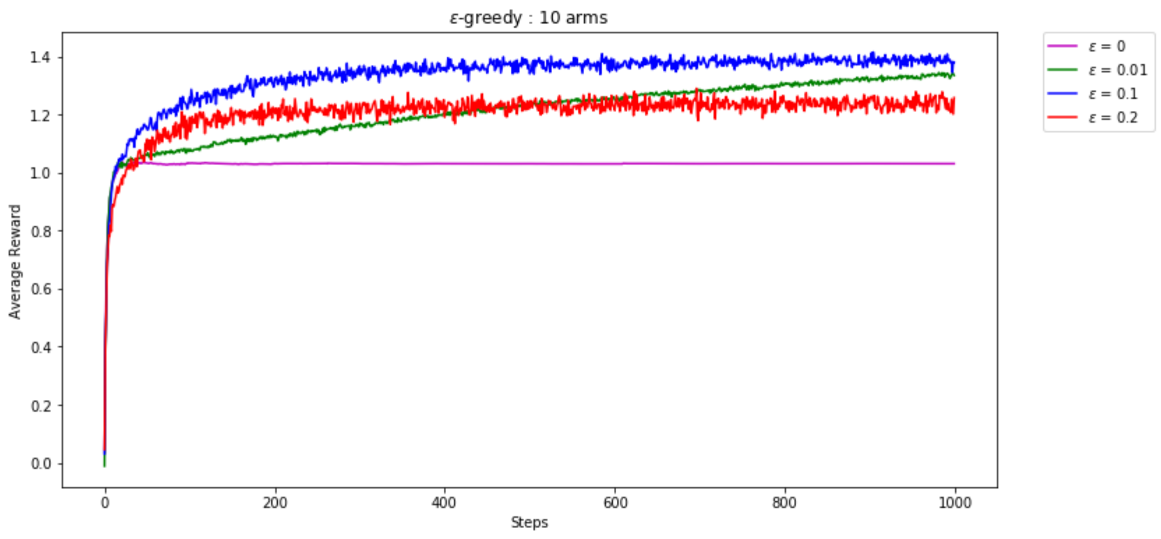

Figure 1: Average performance of ϵ greedy action-value methods on the 10-armed testbed.These data are averages over 2000 runs with different bandit problems.

ϵ = 0 (Greedy): There is no exploration in this case, that is, the program selects only the arm corresponding to maximum reward, whereas there could be an arm which could be potentially better but which never gets selected. This is clearly evident from the low rewards obtained at the end of 1000 steps, compared to ϵ greedy settings.

ϵ = 0.1: This setting outputs the best reward after 1000 steps. There is sufficient exploration and the performance appears to saturate over time.

ϵ = 0.2: Performance slightly less - there is a chance that best arm is not picked due to high exploration.

ϵ = 0.01: The slope of the learning curve indicates that if allowed to run for increased steps, then it can give higher rewards compared to the best case.

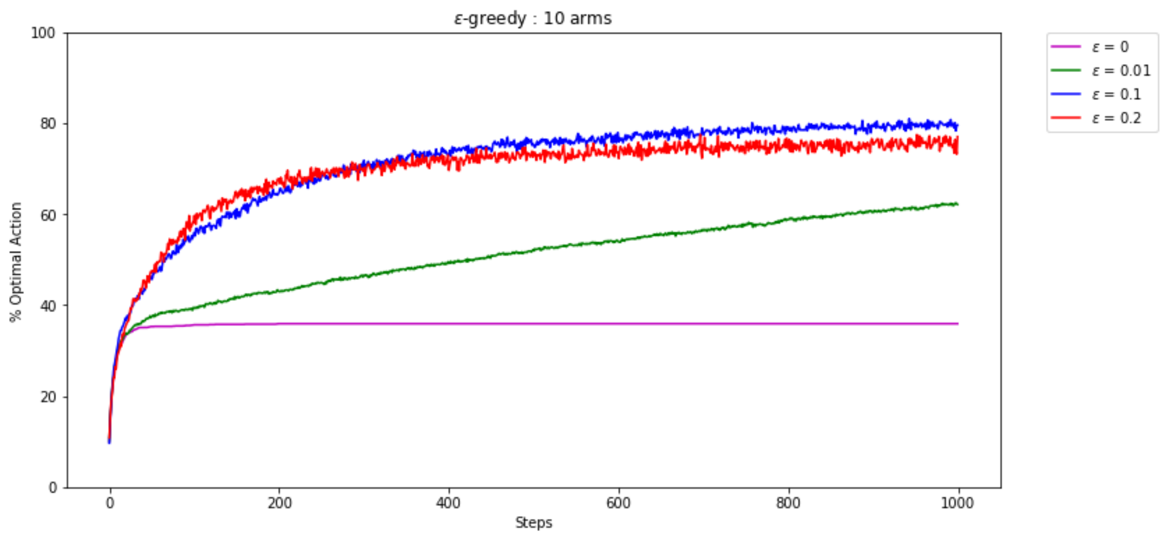

Figure 2: Average number of times optimal action was selected over time

From Figure 2, we observe that the setting with $\epsilon = 0.1/0.2$ achieves 80% accuracy, whereas that of $\epsilon = 0.01$ achieves 60% accuracy. The greedy case finds optimal action for around 35% of the 2000 bandit settings at the end of 1000 time steps.

3 Question 2

- Algorithm: Softmax
- Set of temperatures taken: [0.01,0.1,10,100]

Instead of selecting the best action always, we select actions according to probabilities which in turn are proportional to current estimate.

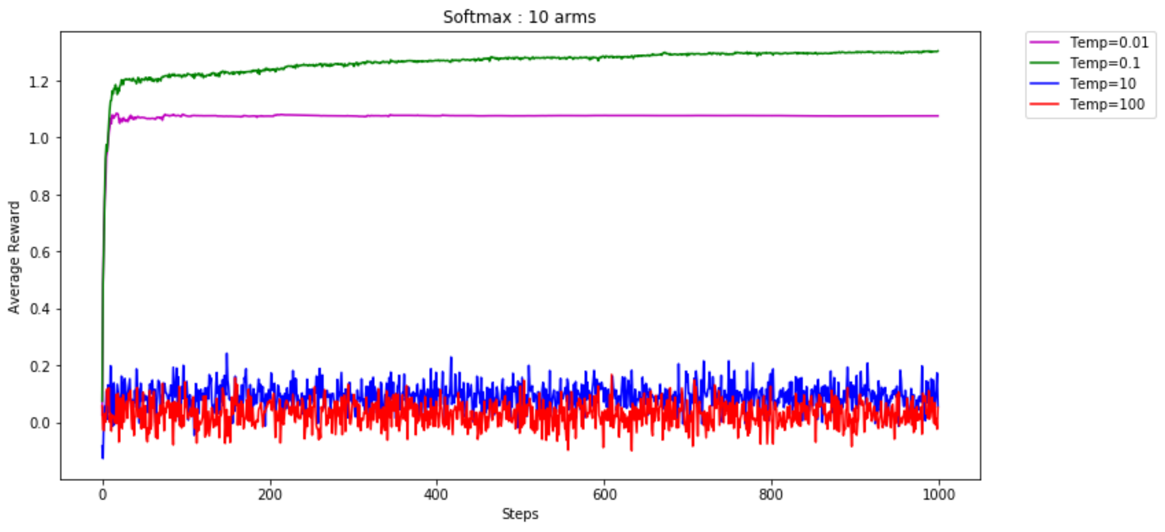

Figure 3: Average Reward Vs Time for Softmax selection criteria

As the temperature increases, the probabilities for picking respective actions tend to become uniform. This implies for a very large Temp = 100, the agent will keep selecting random actions regardless of expectations and hence not much learning takes place as is evident from Figure 3. (In fact, this behavior is observed for Temp=10 or higher)

For low temperatures,say 0.1, the algorithm allots high probabilities to those actions whose expectations are considerably larger. For example, if 2 of 10 actions have comparable rewards (both high), then unlike ϵ greedy which selects only the maximum of the two, the agent in Softmax case selects either of these actions with high probability.

Conversely, if some actions are clearly worse(low expectations), then the probability of picking them is reduced unlike ϵgreedy wherein all actions other than the best are selected with equal chance.

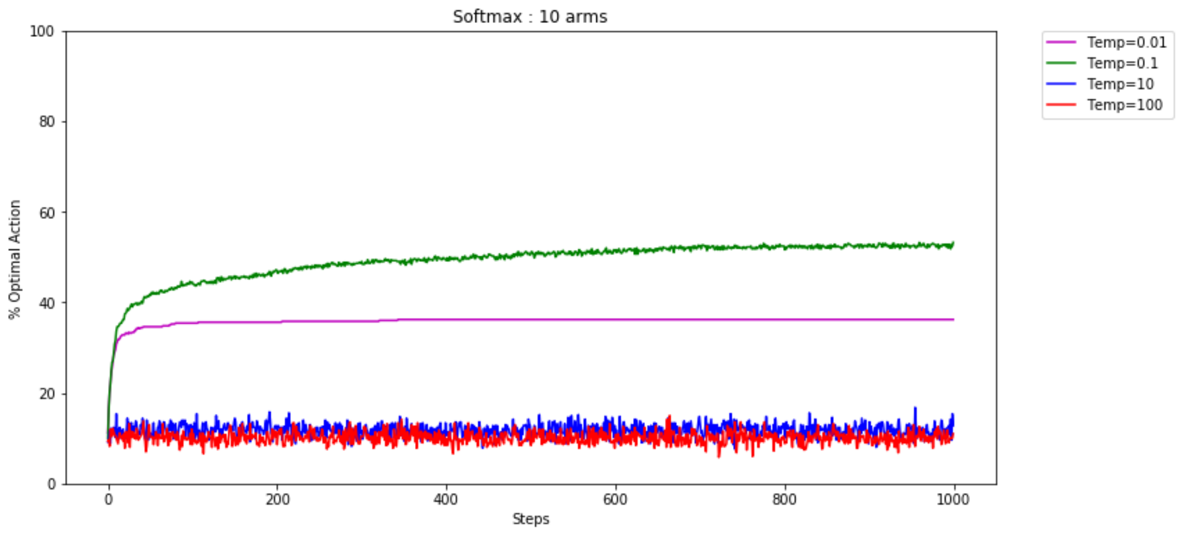

Figure 4: %Optimal Action Selection Vs Time for 2000 settings, Softmax criteria

The case of Temperature =0.1 performs the best. As for optimal action selection, since action selection is not deterministic, optimal will not be selected (only 55% for best setting) as much as in the case of ϵ greedy (80% for best setting), as shown in Figure 4.

4 Question 3

- Algorithm: UCB1
- Setting: c = 2

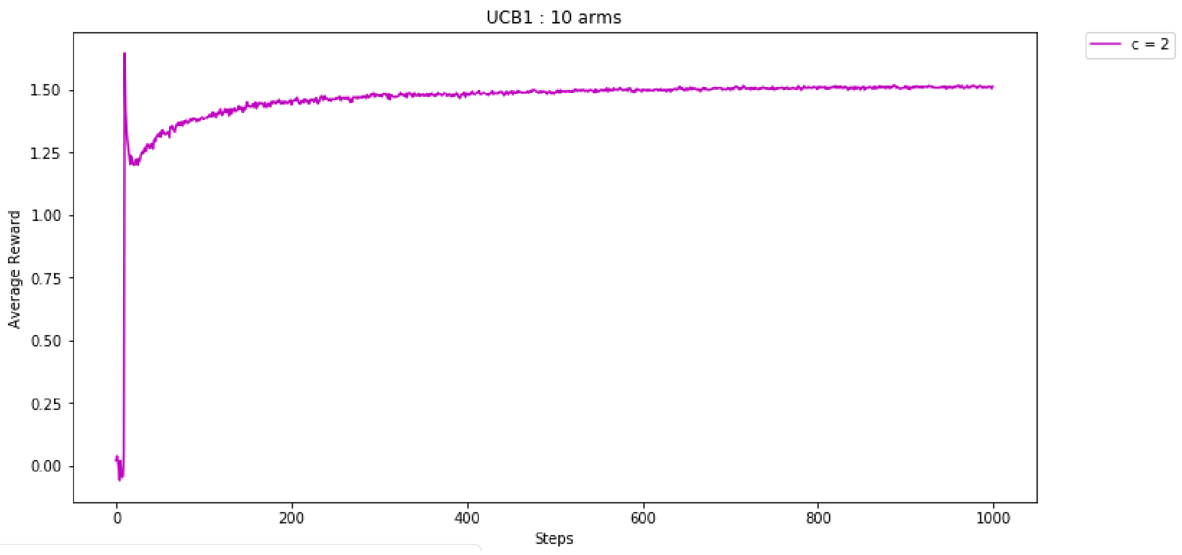

Figure 5: Average Reward Vs Time for UCB1 selection criteria

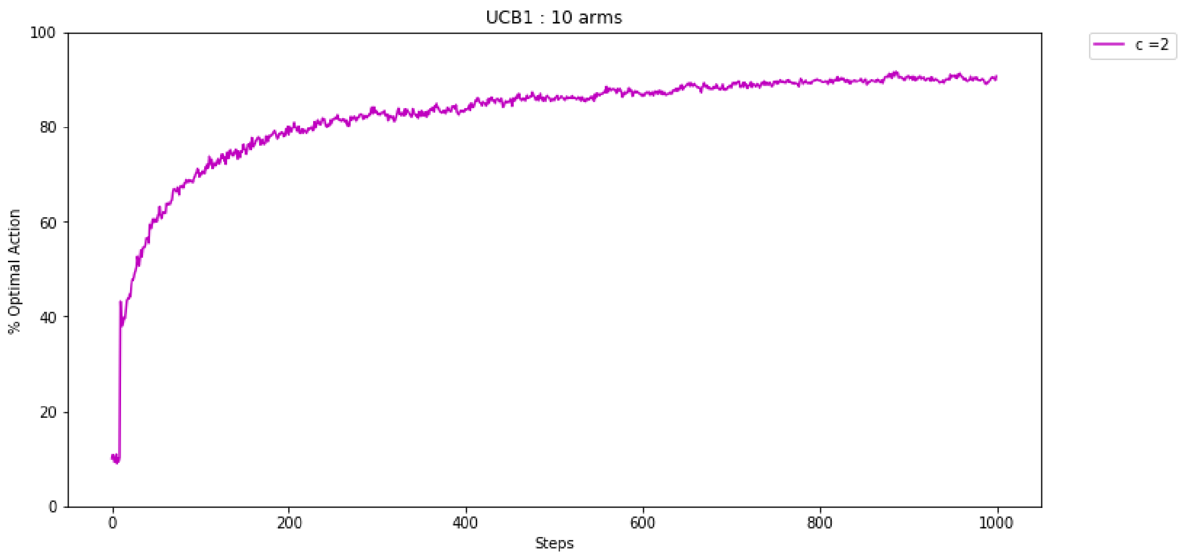

Figure 6: %Optimal Action Selection Vs Time for 2000 settings, UCB1 criteria

In the first 10 time steps, all of 10 actions are selected once. Hence, in the 11th step, the determining factor will be the value Q(a). The spike is due to all agents across all of 2000 runs being in ”sync” to select the best action till that step. But in the next few steps, the agents select different best actions leading to drop in average rewards.

Hence, we observe a distinct spike in performance at the 11th step, and then, decrease in the subsequent steps.

4.1 Comparison Plots

$$A_t \doteq \underset{a}{\operatorname{argmax}} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

For comparison, the best parameter setting (listed in plots) for softmax and ϵ greedy were selected and compared with UCB1. From Figure 7, we observe that UCB1 performs the best among the 3 criteria. Also, ϵ Greedy performs better than softmax.

Inference: Both ϵ greedy and UCB explore well, however, UCB1 selection criteria itself is such that it selects actions not uniform randomly, but according to their potential of actually being optimal, taking into account how close their estimates are to being maximal as well as uncertainties in these estimates. Hence, the exploration policy of UCB1 is better than ϵ Greedy, which is evident from its better performance.

In case of softmax, even though probabilities of selecting best actions are high, there always exists a finite chance of picking actions which are ”worse”. This type of exploration results in less % selection of optimal action and lower performance compared to other algorithms. Refer Figures 7 and 8.

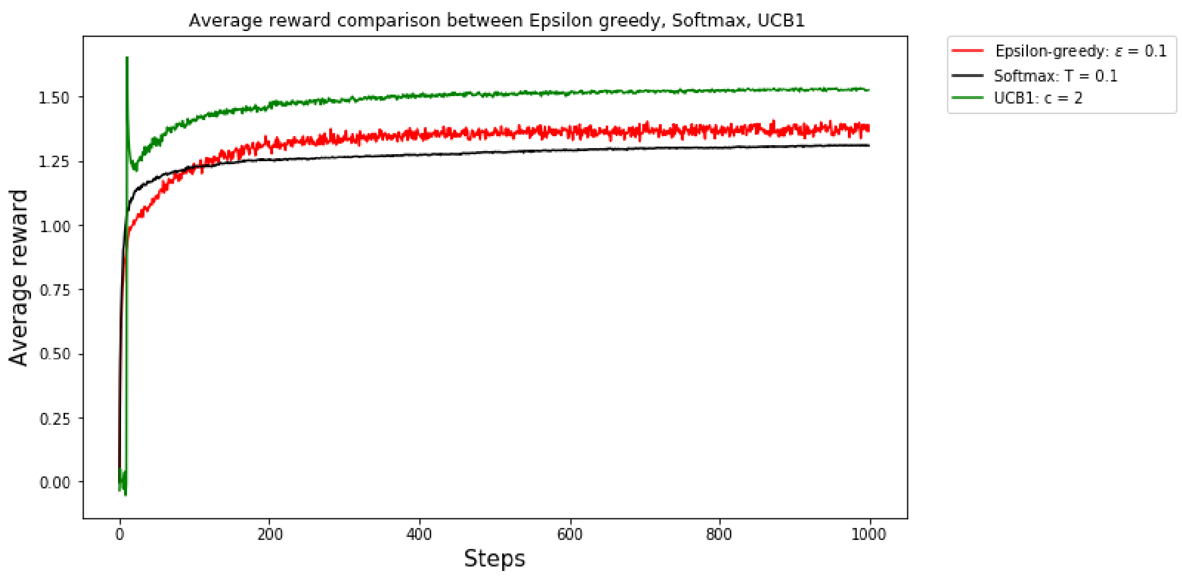

Figure 7: Average Reward Vs Time for comparison between different selection algorithms

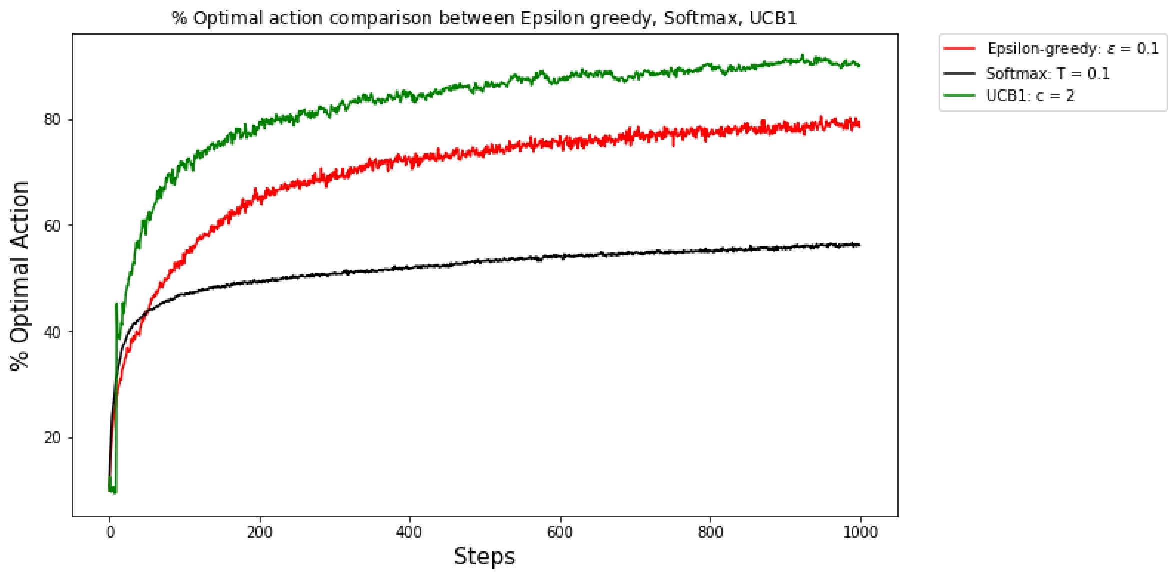

Figure 8: %Optimal Action Selection Vs Time for 2000 settings, comparison between different algorithms

5 NOTE-2

In question 4, Median elimination is run for 200 runs only (instead of 2000 runs as used in the above questions) since it takes an average of 5.5 seconds per run.

6 Question 4

- Algorithm: Median Elimination
- Setting: $\epsilon = 0.1$, $\delta = 0.1$

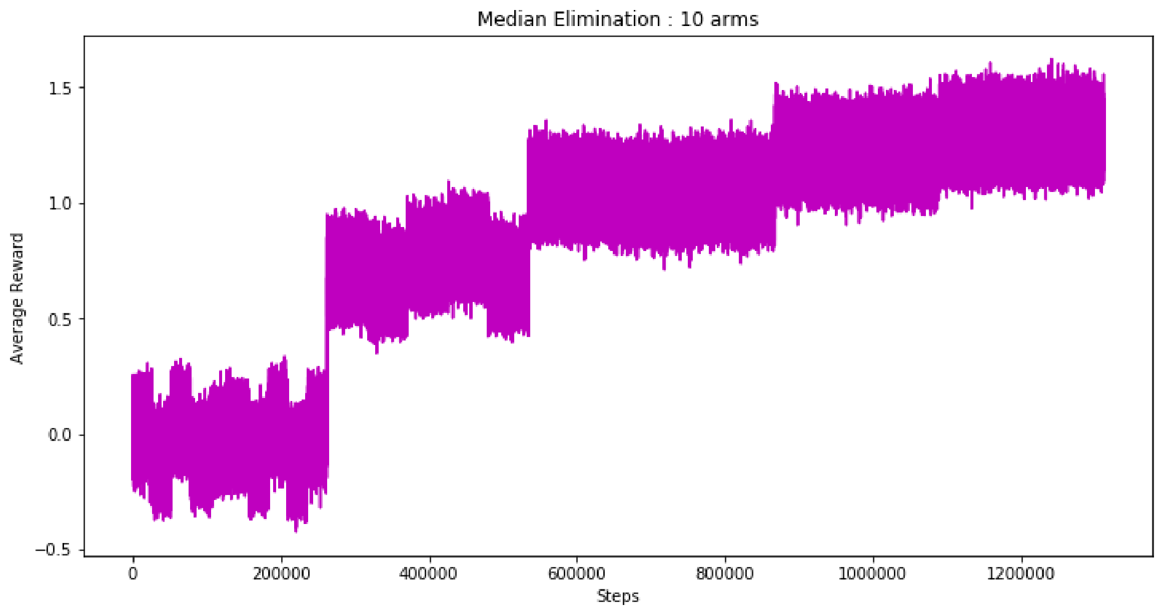

Figure 9: %Average Reward Vs Time for 200 settings, Median Elimination ALgorithm

Median Elimination is more robust compared to UCB, which performed the best among the former 3 algorithms.

6.1 Computational Cost:

The computational cost for computing the median nlog(n), where n is the number of arms. Since this quantity is small for n=10, it is not the rate determining step. However, for large n, say n= 1000, it becomes the rate determining step.

7 Question 5

- Number of arms (k) = 1000
- Number of time steps = 10000
- Number of runs = 2000

As the number of arms increases, the program takes more time to execute. Moreover we need to increase the number of time steps since now, high performance (on par with 10 arm testbed) cannot be achieved with just 1000 time steps. Hence, the number of time steps has been increased to 10000.

Note: Median Elimination plot for 1000 arms has not been generated due to high running time